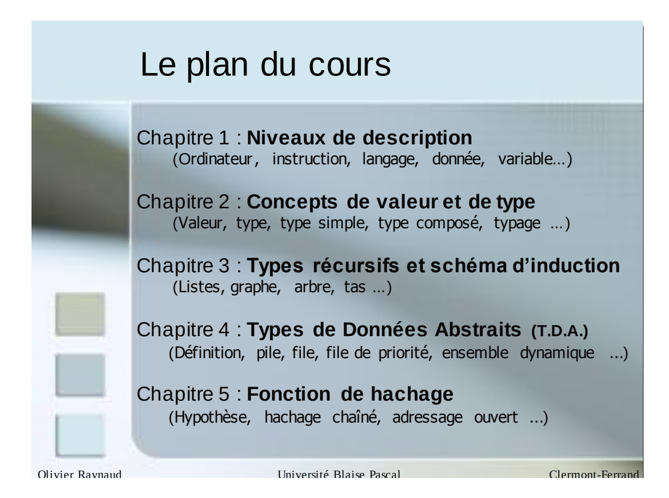
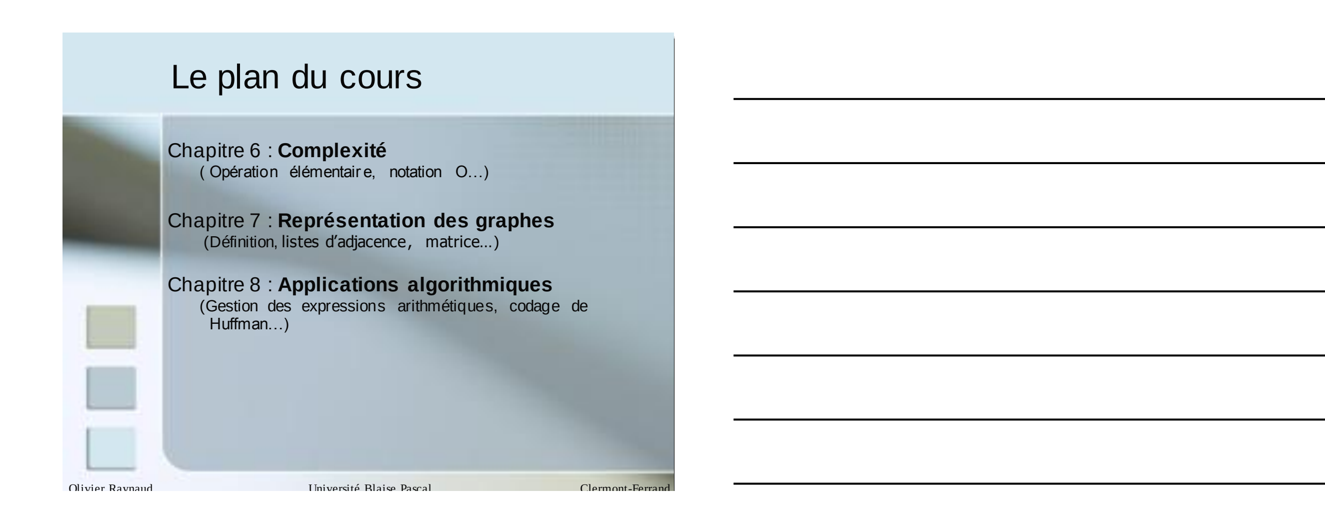

Structures de données
Olivier Raynaud
Université Blaise Pascal


Le plan du cours
Chapitre 1 : Niveaux de description
(Ordinateur, instruction, langage, donnée, variable...)
Chapitre 2 : Concepts de valeur et de type
(Valeur, type, type simple, type composé, typage ...)
Chapitre 3 : Types récursifs et schéma d'induction
(Listes, graphe, arbre, tas ...)
Chapitre 4 : Types de Données Abstraits (T.D.A.)
(Définition, pile, file, file de priorité, ensemble dynamique ...)
Chapitre 5 : Fonction de hachage
(Hypothèse, hachage chaîné, adressage ouvert ...)
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand


Le plan du cours
Chapitre 6 : Complexité
(Opération élémentaire, notation O...)
Chapitre 7 : Représentation des graphes
(Définition, listes d'adjacence, matrice...)
Chapitre 8 : Applications algorithmiques
(Gestion des expressions arithmétiques, codage de Huffman...)
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

Bibliographie

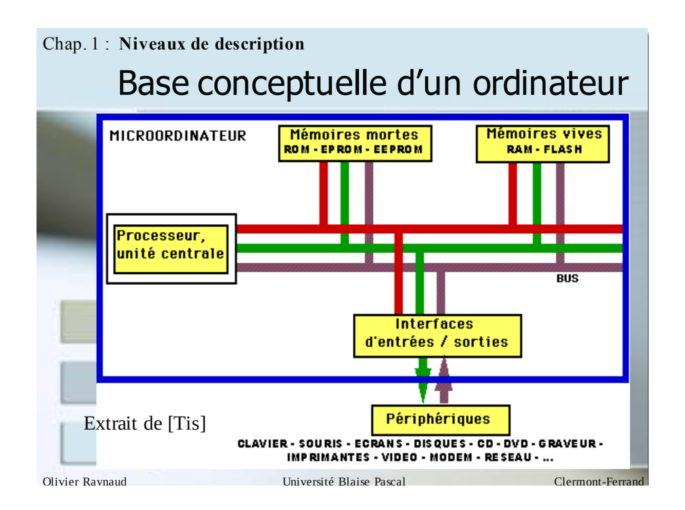
- **L'intelligence et le calcul** *(J.P. Delahaye)* Belin
- [XUO92] **Mathématique discrète et informatique** *(N.H. Xuong)* Masson
- [CLR90] **Introduction à l'algorithmique** *(T. Cormen, C. Leiserson, R. Rivest)* Dunod
- [W90] **Programming Language Concepts and Paradigme** *(David A. Watt)* Prentice Hall
- [KR78] **The C Programming Language** *(B.W. Kernighan and D.M. Ritchie)* Prentice Hall

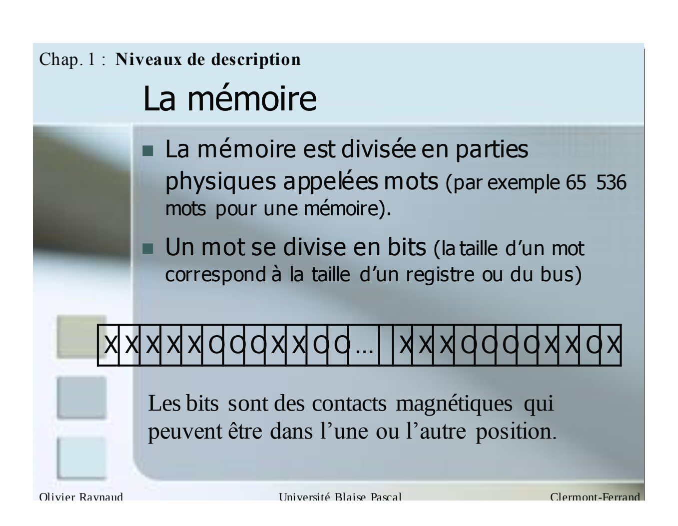
Bibliographie

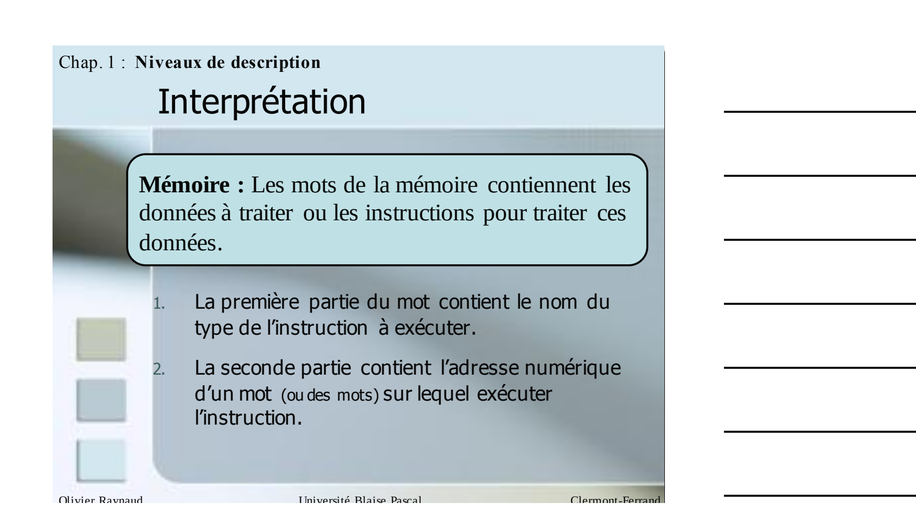
- **Turbo Pascal 4.0 Manuel d'utilisation Borland**
- [GJ00] **Computers and intractability** *(M.R.Garey and D.S.Johnson)* Freeman
- [HOF93] **Godel Escher Bach** *(D.Hofstadter)* InterEdition
- [Ca66] **La logique symbolique** *(L.Caroll)*
- [Tis]w3.mines.unancy.fr/~tisseran/cours/architectures/

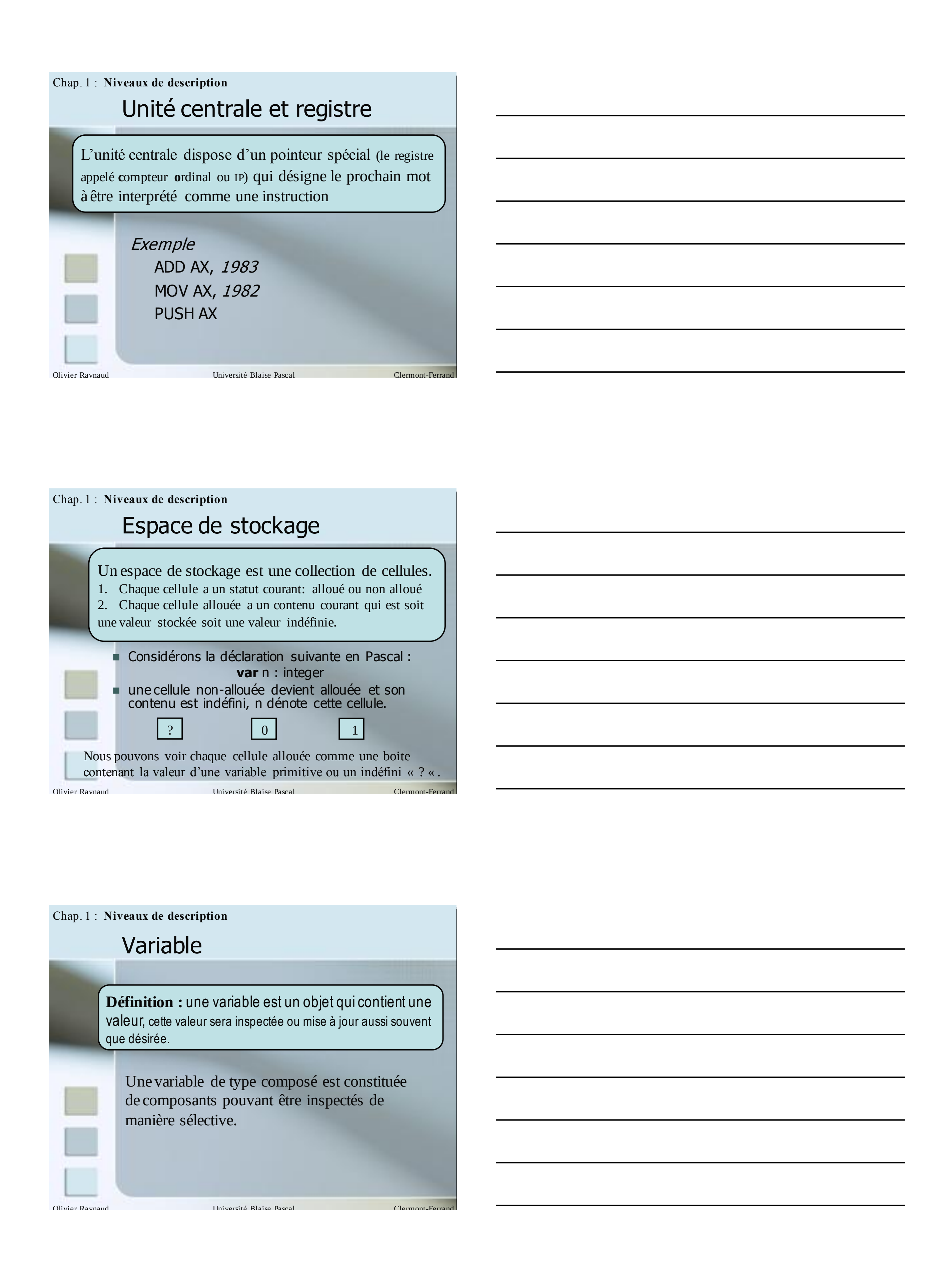
Chapitre 1

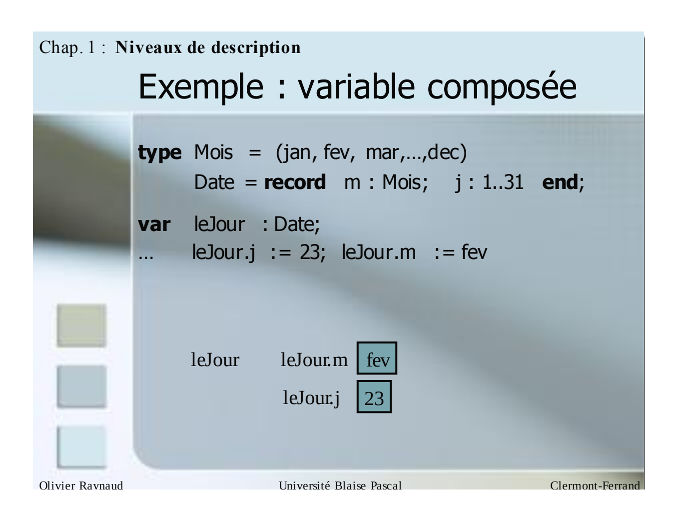
Niveaux de description

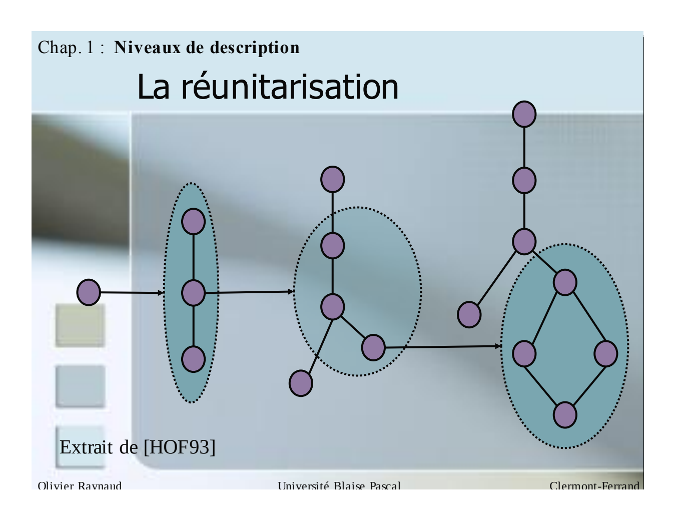

Chap. 1 : Niveaux de description
Base conceptuelle d'un ordinateur
MICROORDINATEUR
Mémoires mortes
ROM - EPROM - EEPROM
Mémoires vives
RAM - FLASH
Processeur,
unité centrale
BUS
Interfaces
d'entrées / sorties
Périphériques
CLAVIER - SOURIS - ECRANS - DISQUES - CD - DVD - GRAVEUR -
IMPRIMANTES - VIDEO - MODEM - RESEAU - ...
Extrait de [Tis]
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

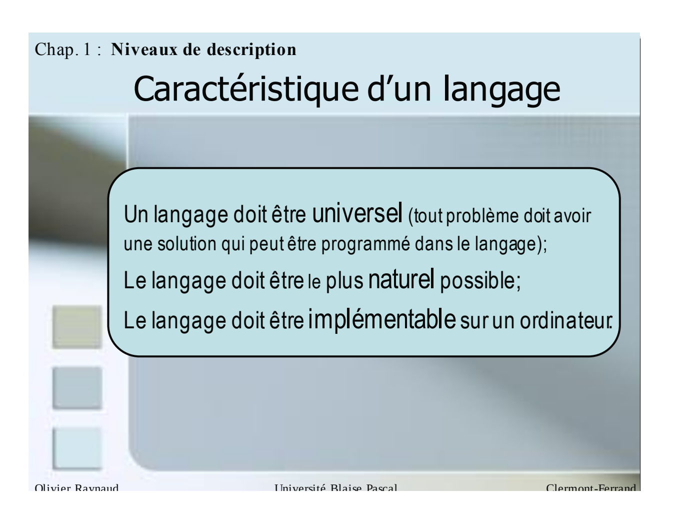

Chap. 1 : Niveaux de description
La mémoire
La mémoire est divisée en parties physiques appelées mots (par exemple 65 536 mots pour une mémoire).
Un mot se divise en bits (la taille d'un mot correspond à la taille d'un registre ou du bus)
x x x x x o o o x x o o ... x x x o o o o x x o x
Les bits sont des contacts magnétiques qui peuvent être dans l'une ou l'autre position.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

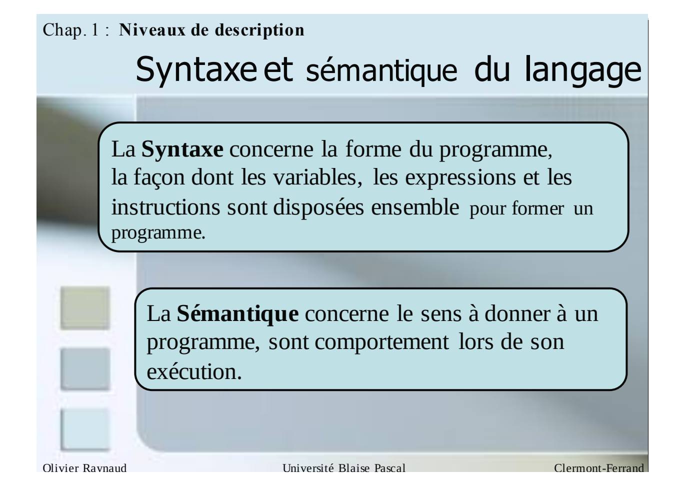

Chap. 1 : Niveaux de description
Interprétation
Mémoire : Les mots de la mémoire contiennent les données à traiter ou les instructions pour traiter ces données.
1. La première partie du mot contient le nom du type de l'instruction à exécuter.
2. La seconde partie contient l'adresse numérique d'un mot (ou des mots) sur lequel exécuter l'instruction.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

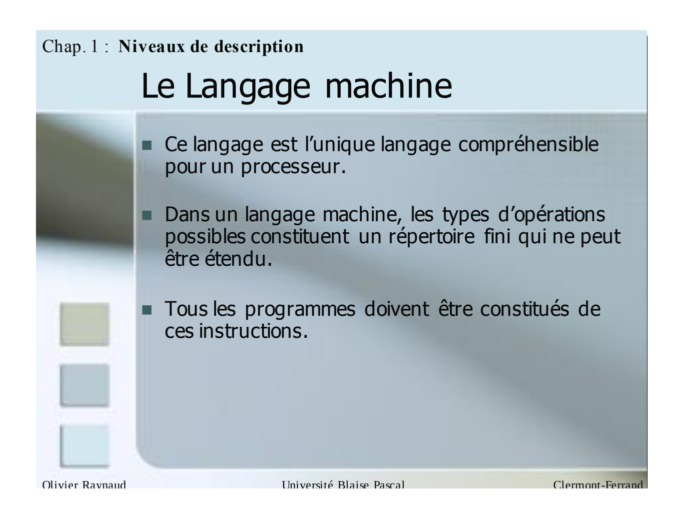

Chap. 1 : Niveaux de description
Unité centrale et registre
L'unité centrale dispose d'un pointeur spécial (le registre appelé compteur ordinal ou IP) qui désigne le prochain mot à être interprété comme une instruction
Exemple
ADD AX, 1983
MOV AX, 1982
PUSH AX
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand
Chap. 1 : Niveaux de description
Espace de stockage
Un espace de stockage est une collection de cellules.
1. Chaque cellule a un statut courant: alloué ou non alloué
2. Chaque cellule allouée a un contenu courant qui est soit une valeur stockée soit une valeur indéfinie.
Considérons la déclaration suivante en Pascal :
var n : integer
une cellule non-allouée devient allouée et son contenu est indéfini, n dénote cette cellule.
?
0
1
Nous pouvons voir chaque cellule allouée comme une boite contenant la valeur d'une variable primitive ou un indéfini « ? « .
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand
Chap. 1 : Niveaux de description
Variable
Définition : une variable est un objet qui contient une valeur, cette valeur sera inspectée ou mise à jour aussi souvent que désirée.
Une variable de type composé est constituée de composants pouvant être inspectés de manière sélective.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

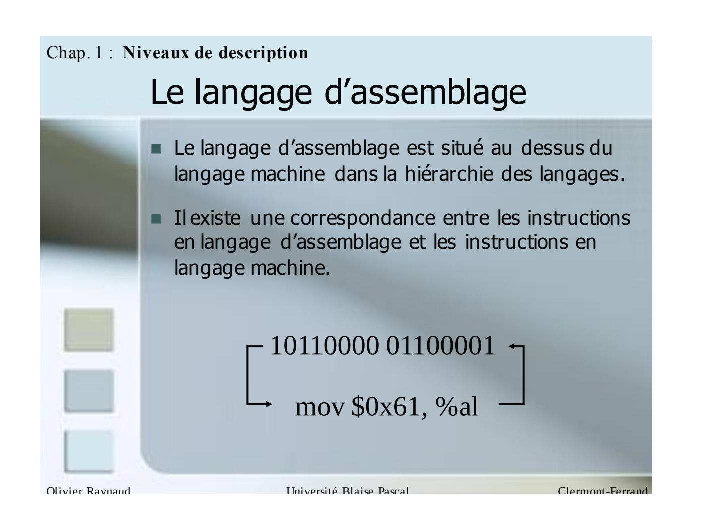

Chap. 1 : Niveaux de description
Exemple : variable composée
type Mois = (jan, fev, mar,…,dec)
Date = record m : Mois; j : 1..31 end;
var leJour : Date;
… leJour.j := 23; leJour.m := fev
leJour
leJour.m
fev
leJour.j
23
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

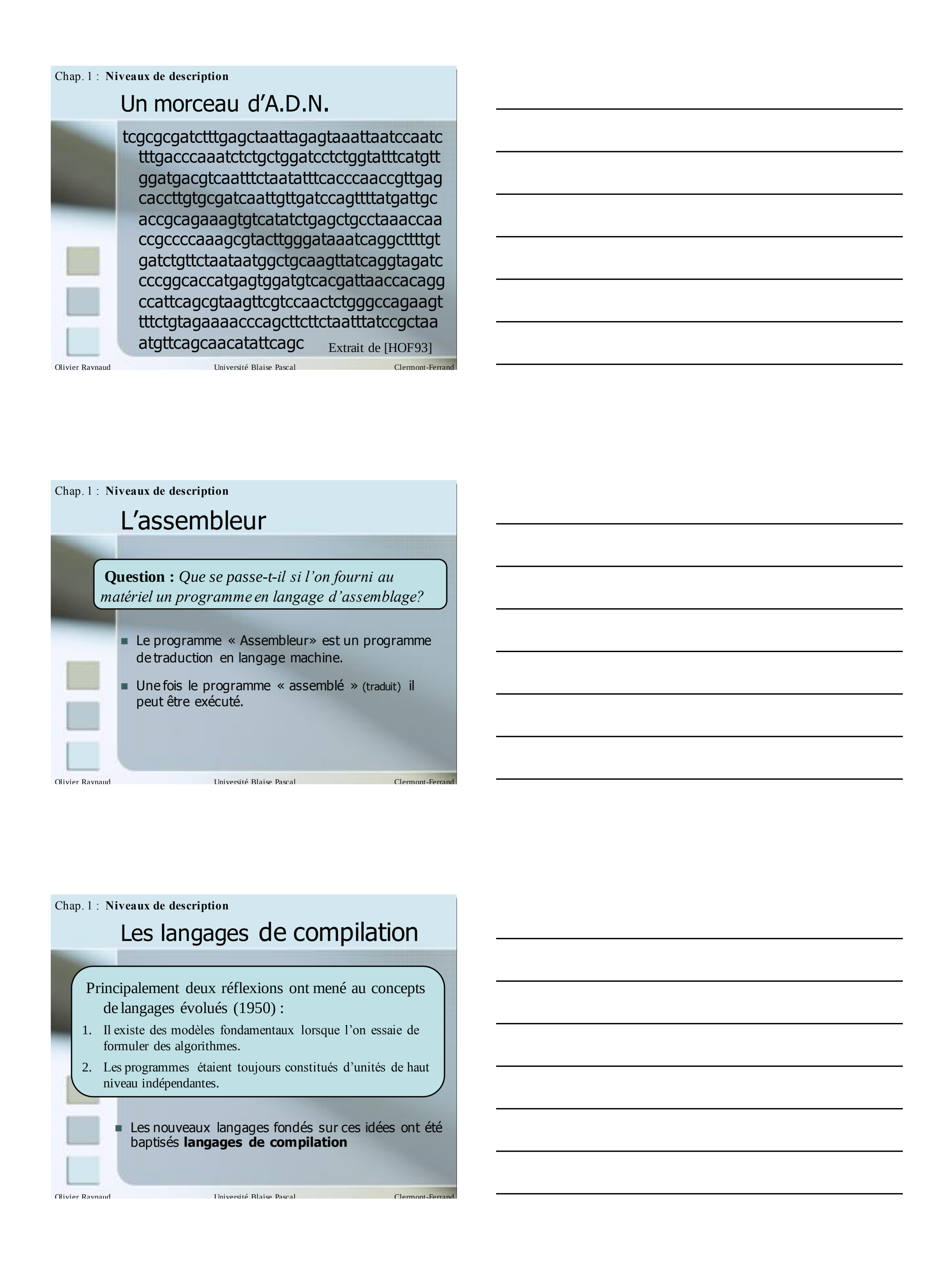

Chap. 1 : Niveaux de description
La réunitarisation
Extrait de [HOF93]
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand


Chap. 1 : Niveaux de description
Caractéristique d'un langage
Un langage doit être universel (tout problème doit avoir une solution qui peut être programmé dans le langage);
Le langage doit être le plus naturel possible;
Le langage doit être implémentable sur un ordinateur.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

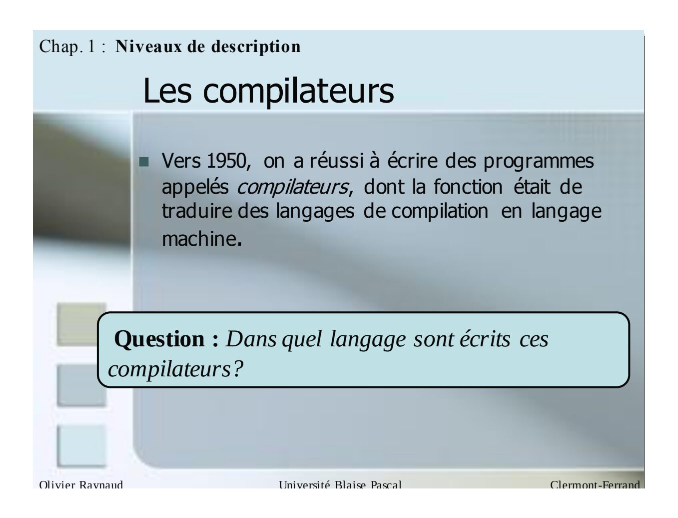

Chap. 1 : Niveaux de description
Syntaxe et sémantique du langage
La Syntaxe concerne la forme du programme, la façon dont les variables, les expressions et les instructions sont disposées ensemble pour former un programme.
La Sémantique concerne le sens à donner à un programme, sont comportement lors de son exécution.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

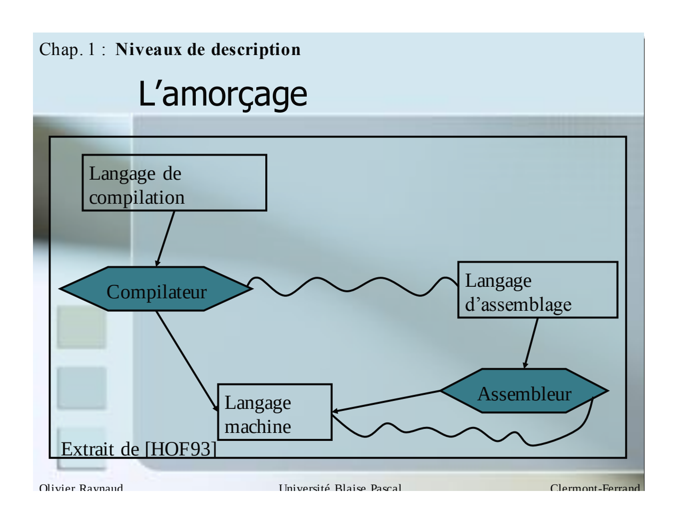

Chap. 1 : Niveaux de description
Le Langage machine
Ce langage est l'unique langage compréhensible pour un processeur.
Dans un langage machine, les types d'opérations possibles constituent un répertoire fini qui ne peut être étendu.
Tous les programmes doivent être constitués de ces instructions.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand


Chap. 1 : Niveaux de description
Le langage d'assemblage
Le langage d'assemblage est situé au dessus du langage machine dans la hiérarchie des langages.
Il existe une correspondance entre les instructions en langage d'assemblage et les instructions en langage machine.
10110000 01100001
mov $0x61, %al
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

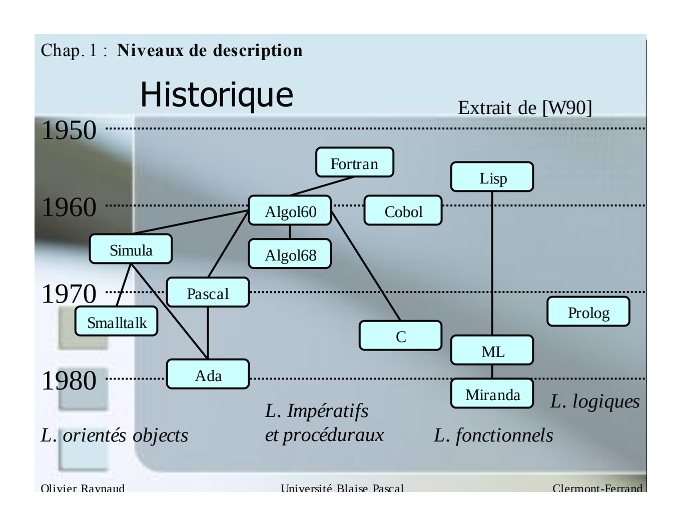

Chap. 1 : Niveaux de description
Un morceau d'A.D.N.
tcgcgcgatctttgagctaattagagtaaattaatccaatc tttgacccaaatctctgctggatcctctggtatttcatgtt ggatgacgtcaatttctaatatttcacccaaccgttgag caccttgtgcgatcaattgttgatccagttttatgattgc accgcagaaagtgtcatatctgagctgcctaaaccaa ccgccccaaagcgtacttgggataaatcaggcttttgt gatctgttctaataatggctgcaagttatcaggtagatc cccggcaccatgagtggatgtcacgattaaccacagg ccattcagcgtaagttcgtccaactctgggccagaagt tttctgtagaaaacccagcttcttctaatttatccgctaa atgttcagcaacatattcagc
Extrait de [HOF93]
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand
Chap. 1 : Niveaux de description
L'assembleur
Question : Que se passe-t-il si l'on fourni au matériel un programme en langage d'assemblage?
Le programme « Assembleur» est un programme de traduction en langage machine.
Une fois le programme « assemblé » (traduit) il peut être exécuté.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand
Chap. 1 : Niveaux de description
Les langages de compilation
Principalement deux réflexions ont mené au concepts de langages évolués (1950) :
1. Il existe des modèles fondamentaux lorsque l'on essaie de formuler des algorithmes.
2. Les programmes étaient toujours constitués d'unités de haut niveau indépendantes.
Les nouveaux langages fondés sur ces idées ont été baptisés langages de compilation
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

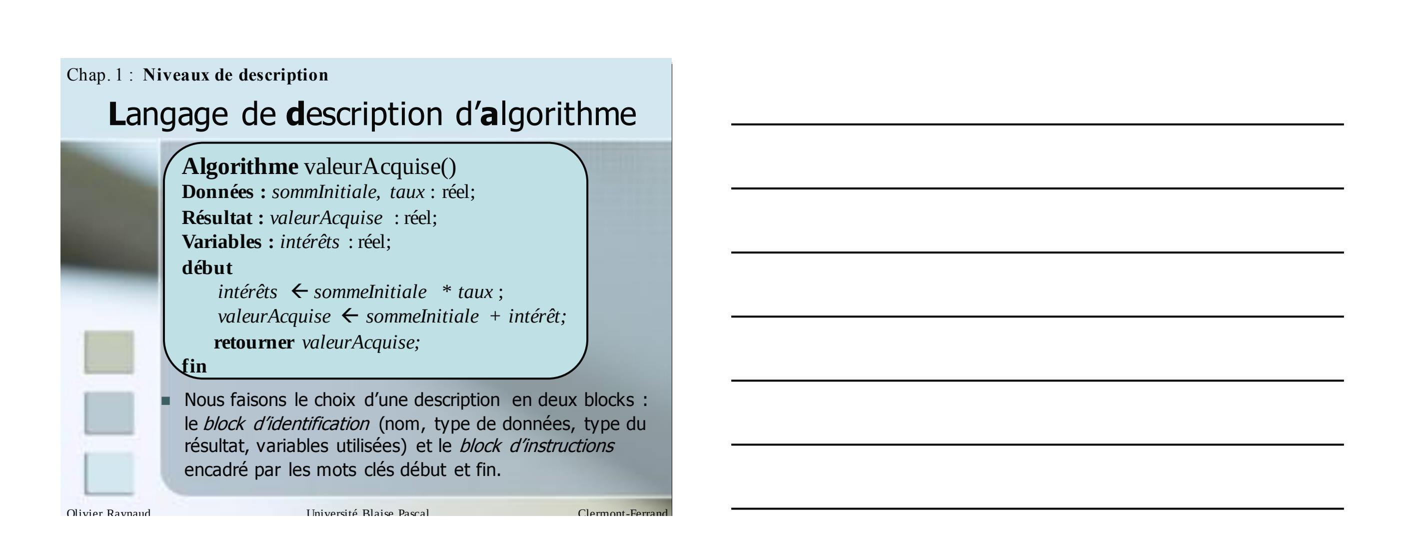

Chap. 1 : Niveaux de description
Les trois niveaux de description
Niveau en langage machine:
« Exécution du programme interrompue au point 1110010101110111 »
Niveau en langage d'assemblage:
« Exécution du programme interrompue lorsque l'instruction DIV (division) a été rencontrée »
Niveau en langage de compilation:
« Exécution du programme interrompue lors de l'examen de l'expression algébrique « (A+B)/Z». »
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

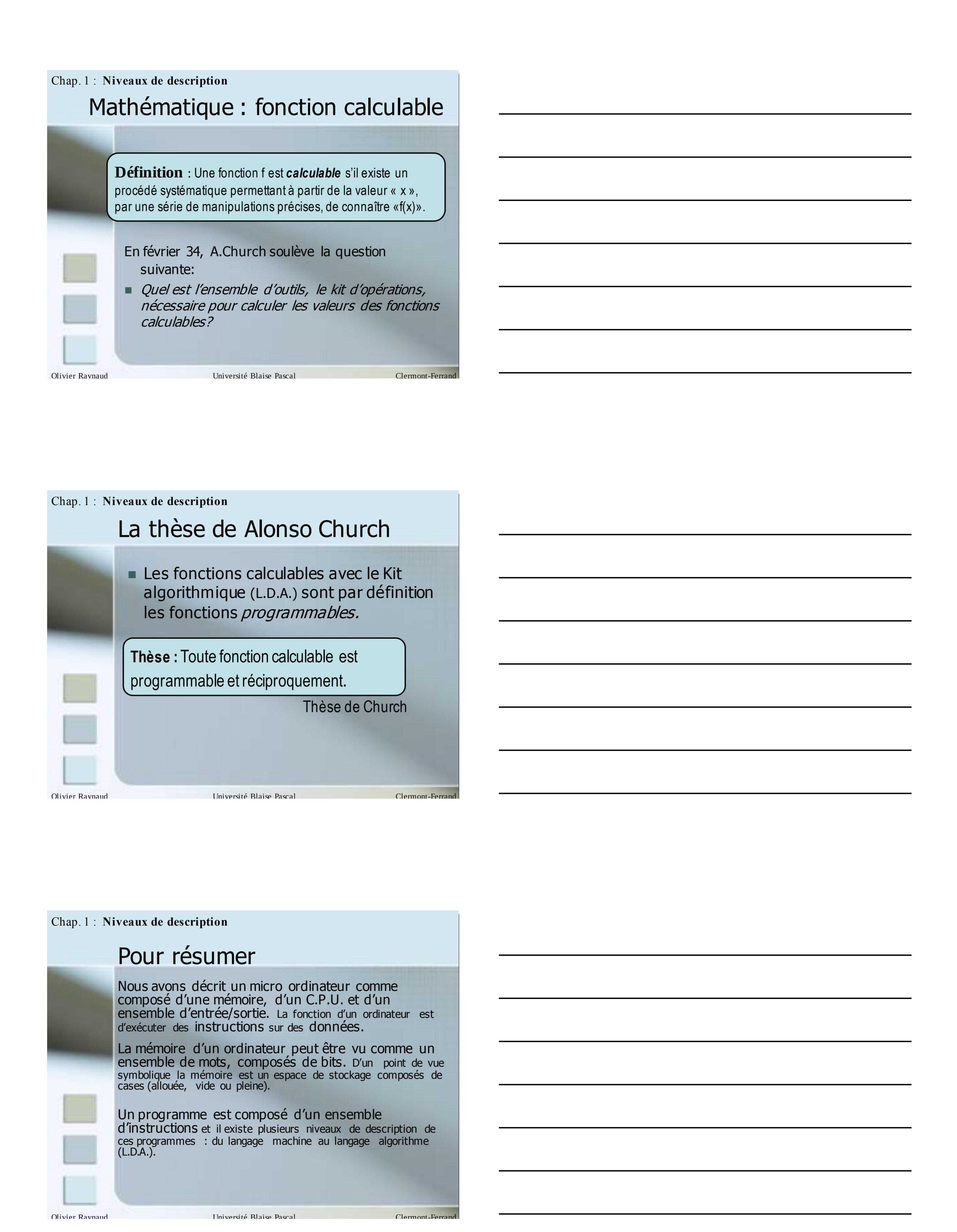

Chap. 1 : Niveaux de description
Les compilateurs
Vers 1950, on a réussi à écrire des programmes appelés compilateurs, dont la fonction était de traduire des langages de compilation en langage machine.
Question : Dans quel langage sont écrits ces compilateurs?
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

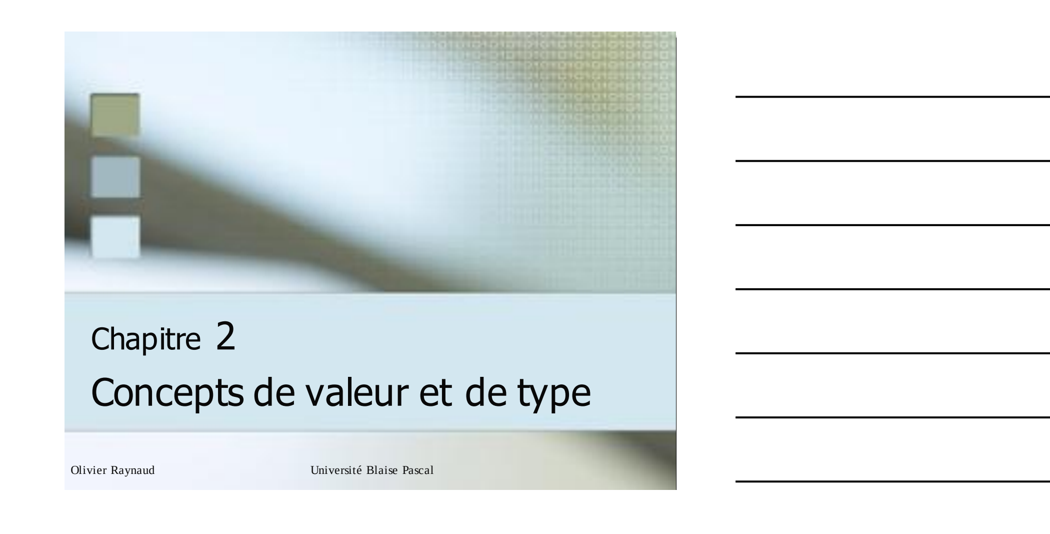

Chap. 1 : Niveaux de description
L'amorçage
Langage de compilation
Compilateur
Langage d'assemblage
Assembleur
Langage machine
Extrait de [HOF93]
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

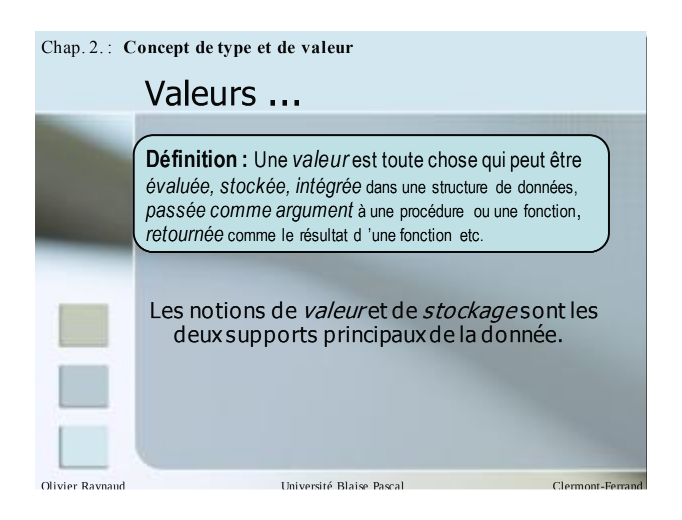

Chap. 1 : Niveaux de description
Les interpréteurs
Ils assurent la traduction des langages évolués en langage machine en lisant un programme ligne à ligne et en exécutant immédiatement cette ligne.
Un interpréteur est donc au compilateur ce qu'un interprète simultané est à un traducteur.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

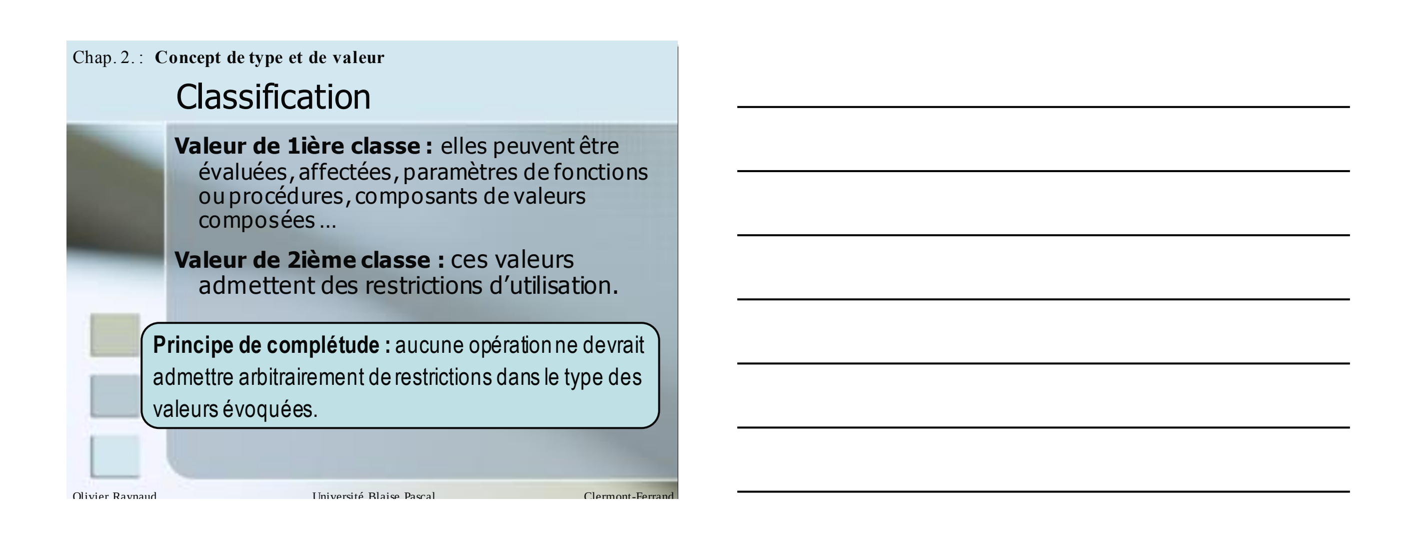

Chap. 1 : Niveaux de description
Historique
Extrait de [W90]
1950
1960
1970
1980
Fortran
Lisp
Algol60
Cobol
Simula
Algol68
Pascal
Prolog
Smalltalk
C
ML
Ada
Miranda
L. Impératifs et procéduraux
L. logiques
L. orientés objects
L. fonctionnels
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

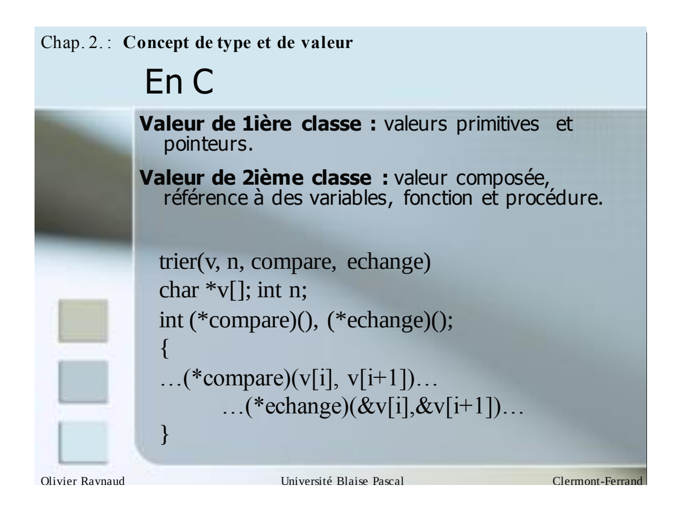

Chap. 1 : Niveaux de description
Langage de description d'algorithme
Algorithme valeurAcquise()
Données : sommInitiale, taux : réel;
Résultat : valeurAcquise : réel;
Variables : intérêts : réel;
début
intérêts ← sommeInitiale * taux ;
valeurAcquise ← sommeInitiale + intérêt;
retourner valeurAcquise;
fin
Nous faisons le choix d'une description en deux blocks : le block d'identification (nom, type de données, type du résultat, variables utilisées) et le block d'instructions encadré par les mots clés début et fin.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

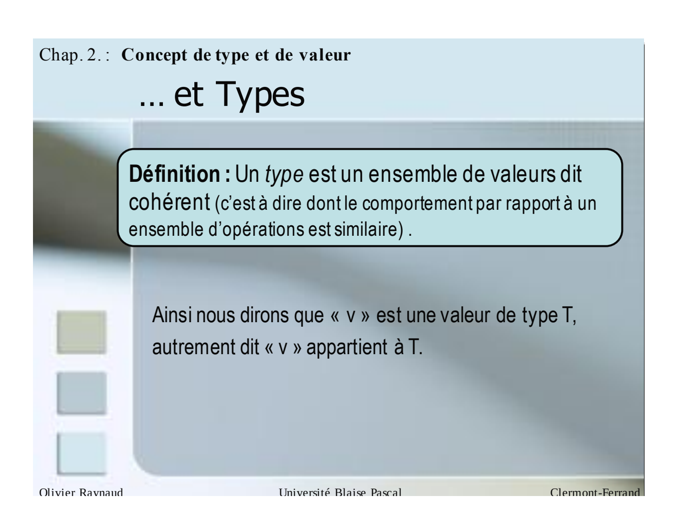

Chap. 1 : Niveaux de description
Mathématique : fonction calculable
Définition : Une fonction f est calculable s'il existe un procédé systématique permettant à partir de la valeur « x », par une série de manipulations précises, de connaître «f(x)».
En février 34, A.Church soulève la question suivante:
Quel est l'ensemble d'outils, le kit d'opérations, nécessaire pour calculer les valeurs des fonctions calculables?
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand
Chap. 1 : Niveaux de description
La thèse de Alonso Church
Les fonctions calculables avec le Kit algorithmique (L.D.A.) sont par définition les fonctions programmables.
Thèse : Toute fonction calculable est programmable et réciproquement.
Thèse de Church
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand
Chap. 1 : Niveaux de description
Pour résumer
Nous avons décrit un micro ordinateur comme composé d'une mémoire, d'un C.P.U. et d'un ensemble d'entrée/sortie. La fonction d'un ordinateur est d'exécuter des instructions sur des données.
La mémoire d'un ordinateur peut être vu comme un ensemble de mots, composés de bits. D'un point de vue symbolique la mémoire est un espace de stockage composés de cases (allouée, vide ou pleine).
Un programme est composé d'un ensemble d'instructions et il existe plusieurs niveaux de description de ces programmes : du langage machine au langage algorithme (L.D.A.).
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

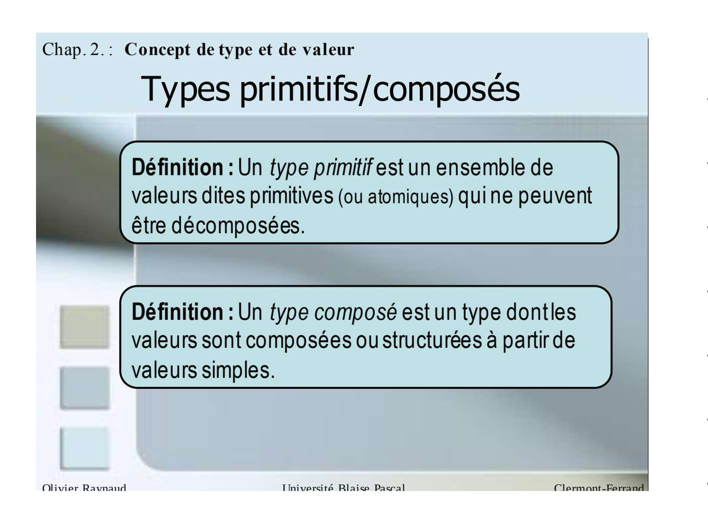

Chapitre 2
Concepts de valeur et de type
Olivier Raynaud
Université Blaise Pascal

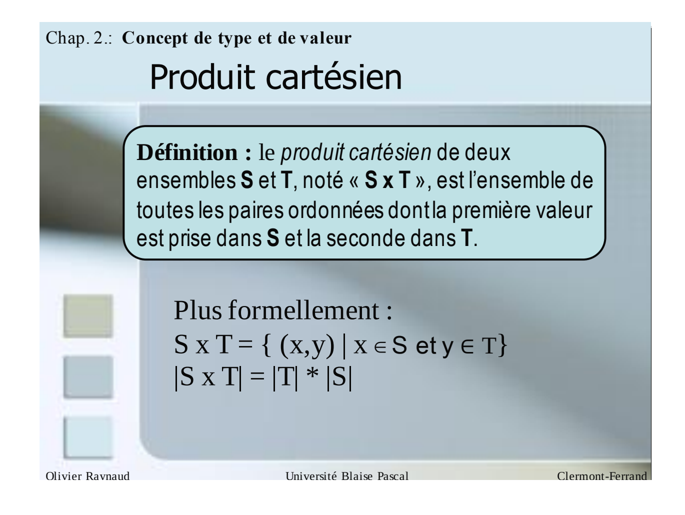

Chap. 2. : Concept de type et de valeur
Valeurs ...
Définition : Une valeur est toute chose qui peut être évaluée, stockée, intégrée dans une structure de données, passée comme argument à une procédure ou une fonction, retournée comme le résultat d 'une fonction etc.
Les notions de valeur et de stockage sont les deux supports principaux de la donnée.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand


Chap. 2. : Concept de type et de valeur
Classification
Valeur de 1ière classe : elles peuvent être évaluées, affectées, paramètres de fonctions ou procédures, composants de valeurs composées ...
Valeur de 2ième classe : ces valeurs admettent des restrictions d'utilisation.
Principe de complétude : aucune opération ne devrait admettre arbitrairement de restrictions dans le type des valeurs évoquées.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

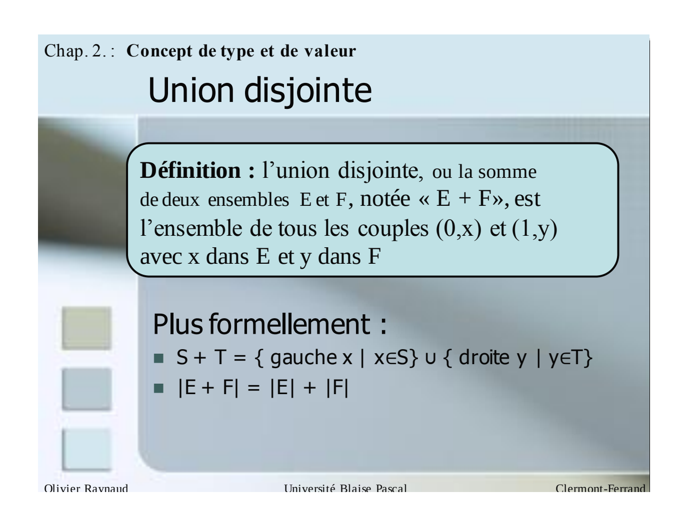

Chap. 2. : Concept de type et de valeur
En C
Valeur de 1ière classe : valeurs primitives et pointeurs.
Valeur de 2ième classe : valeur composée, référence à des variables, fonction et procédure.
trier(v, n, compare, echange)
char *v[]; int n;
int (*compare)(), (*echange)();
{
…(*compare)(v[i], v[i+1])…
…(*echange)(&v[i],&v[i+1])…
}
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand


Chap. 2. : Concept de type et de valeur
… et Types
Définition : Un type est un ensemble de valeurs dit cohérent (c'est à dire dont le comportement par rapport à un ensemble d'opérations est similaire) .
Ainsi nous dirons que « v » est une valeur de type T, autrement dit « v » appartient à T.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

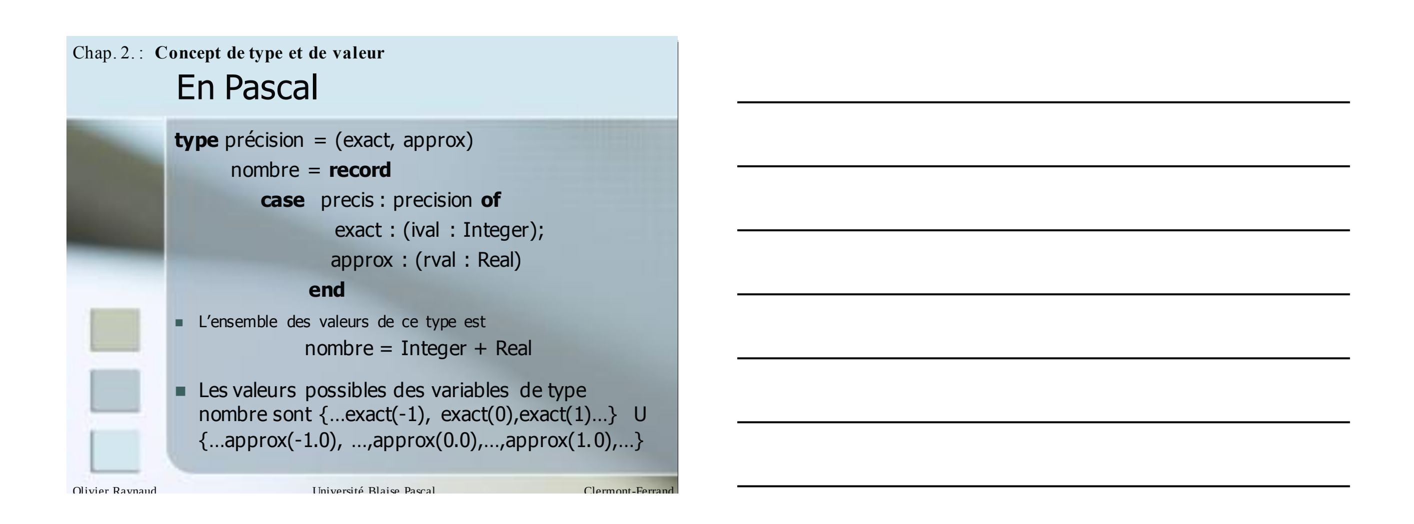

Chap. 2. : Concept de type et de valeur
Types primitifs/composés
Définition : Un type primitif est un ensemble de valeurs dites primitives (ou atomiques) qui ne peuvent être décomposées.
Définition : Un type composé est un type dont les valeurs sont composées ou structurées à partir de valeurs simples.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand



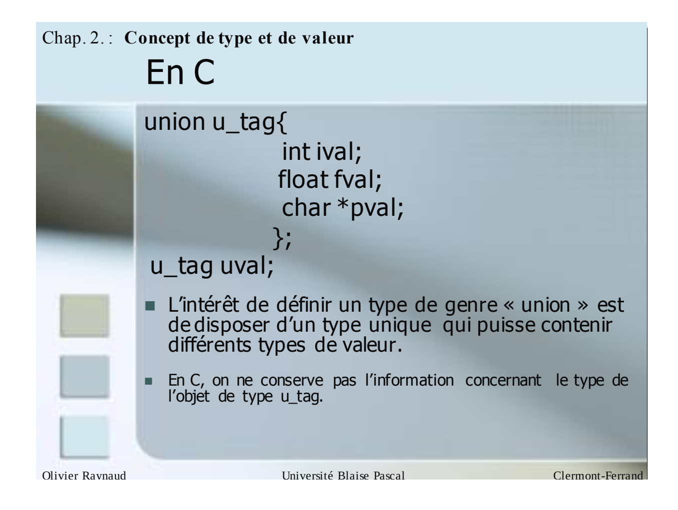
Chap. 2.: Concept de type et de valeur
Produit cartésien
Définition : le produit cartésien de deux ensembles S et T, noté « S x T », est l'ensemble de toutes les paires ordonnées dont la première valeur est prise dans S et la seconde dans T.
Plus formellement :
S x T = { (x,y) | x ∈ S et y ∈ T}
|S x T| = |T| * |S|
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

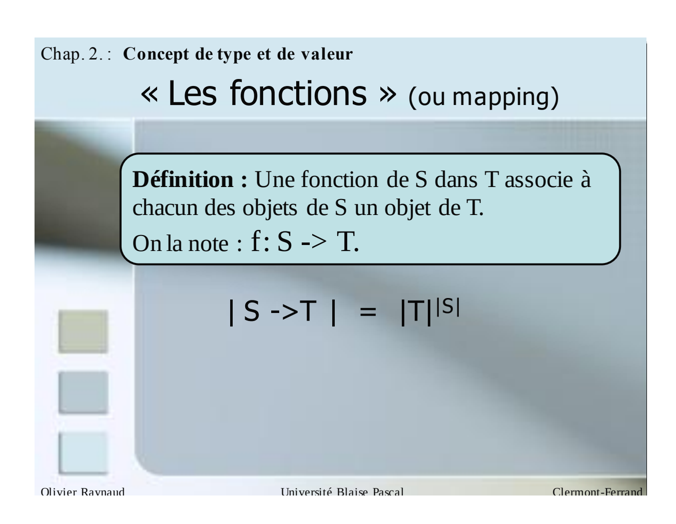


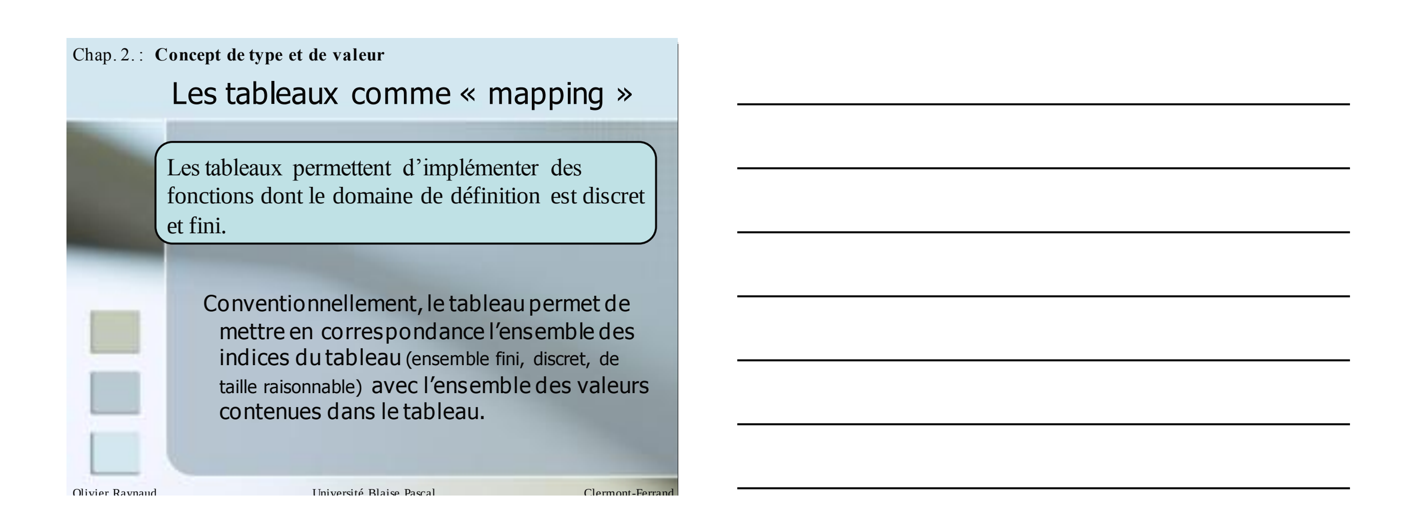
Chap. 2. : Concept de type et de valeur
Union disjointe
Définition : l'union disjointe, ou la somme de deux ensembles E et F, notée « E + F», est l'ensemble de tous les couples (0,x) et (1,y) avec x dans E et y dans F
Plus formellement :
S + T = { gauche x | x∈S} ∪ { droite y | y∈T}
|E + F| = |E| + |F|
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

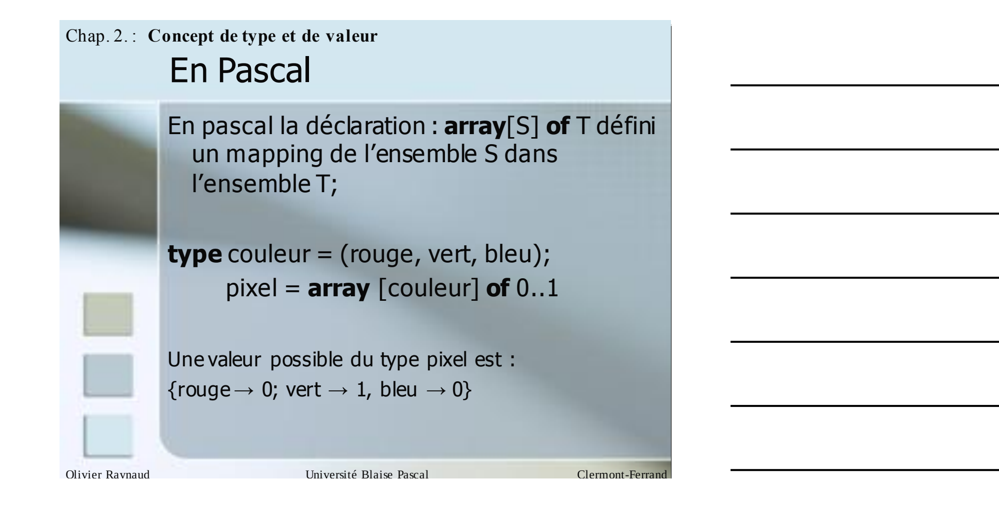


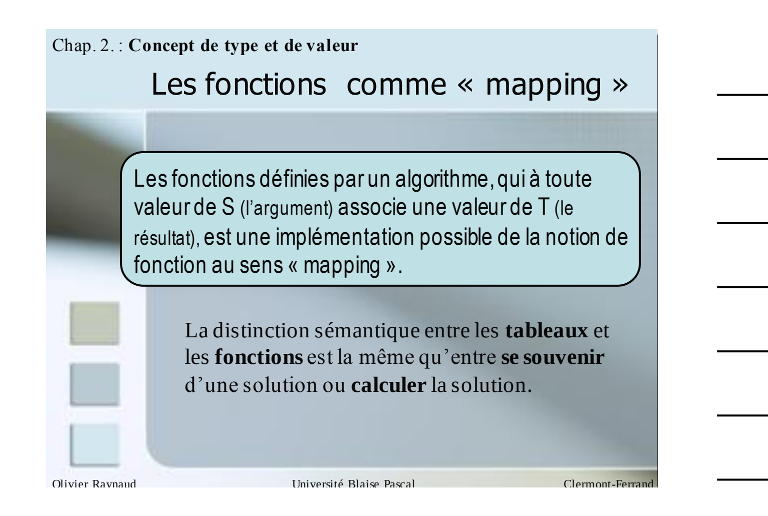
Chap. 2. : Concept de type et de valeur
En Pascal
type précision = (exact, approx)
nombre = record
case precis : precision of
exact : (ival : Integer);
approx : (rval : Real)
end
L'ensemble des valeurs de ce type est
nombre = Integer + Real
Les valeurs possibles des variables de type nombre sont {…exact(-1), exact(0),exact(1)…} U {…approx(-1.0), …,approx(0.0),…,approx(1.0),…}
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

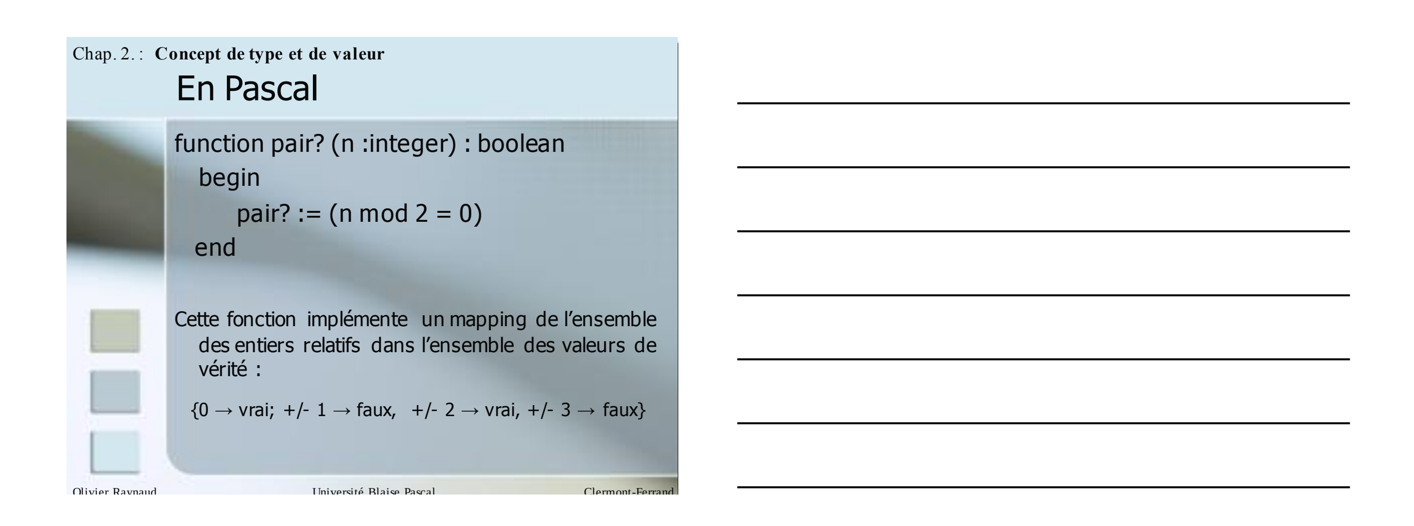

Chap. 2. : Concept de type et de valeur
En C
union u_tag{
int ival;
float fval;
char *pval;
};
u_tag uval;
L'intérêt de définir un type de genre « union » est de disposer d'un type unique qui puisse contenir différents types de valeur.
En C, on ne conserve pas l'information concernant le type de l'objet de type u_tag.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

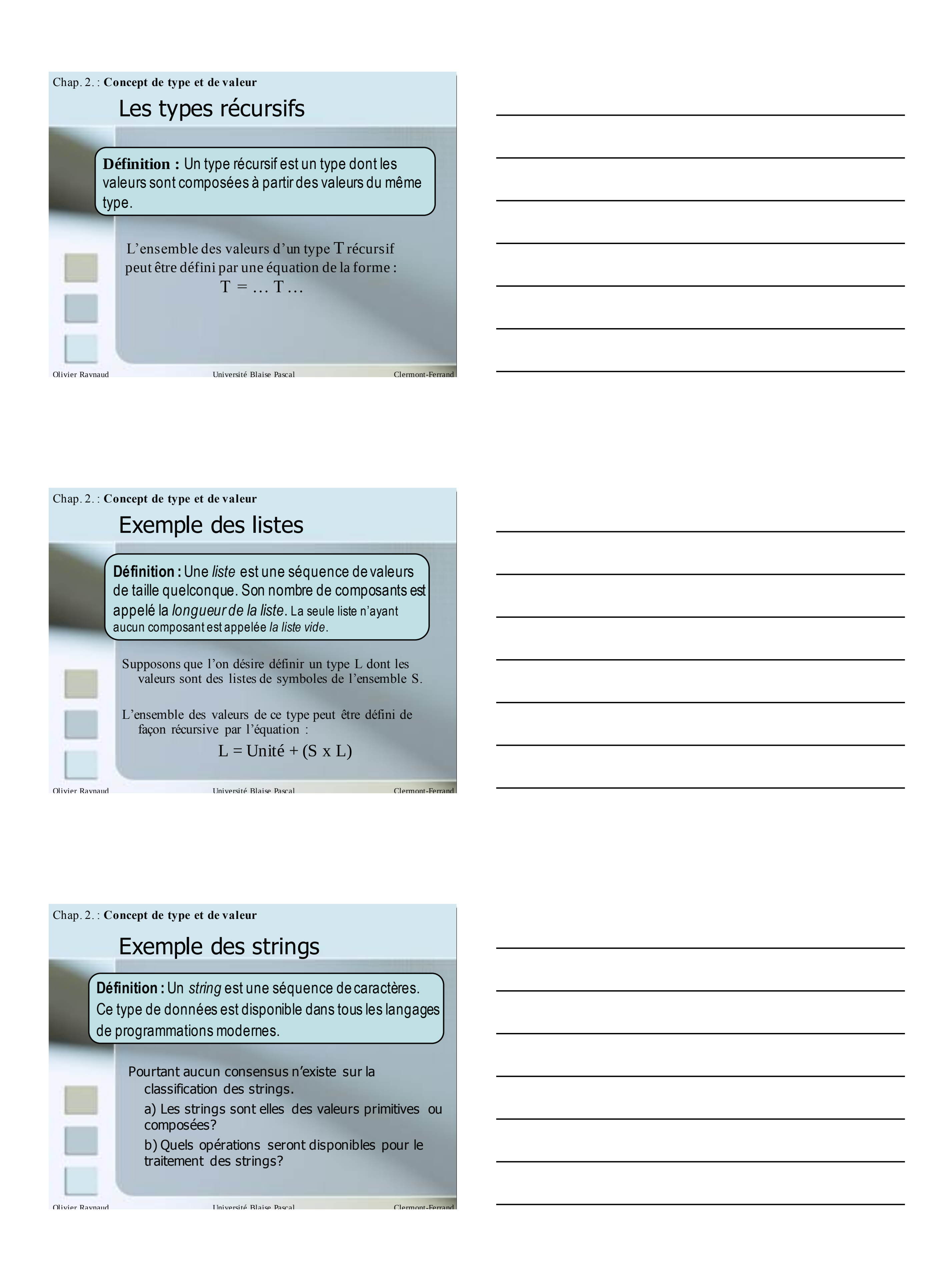

Chap. 2. : Concept de type et de valeur
« Les fonctions » (ou mapping)
Définition : Une fonction de S dans T associe à chacun des objets de S un objet de T.
On la note : f: S -> T.
| S ->T | = |T|^|S|
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

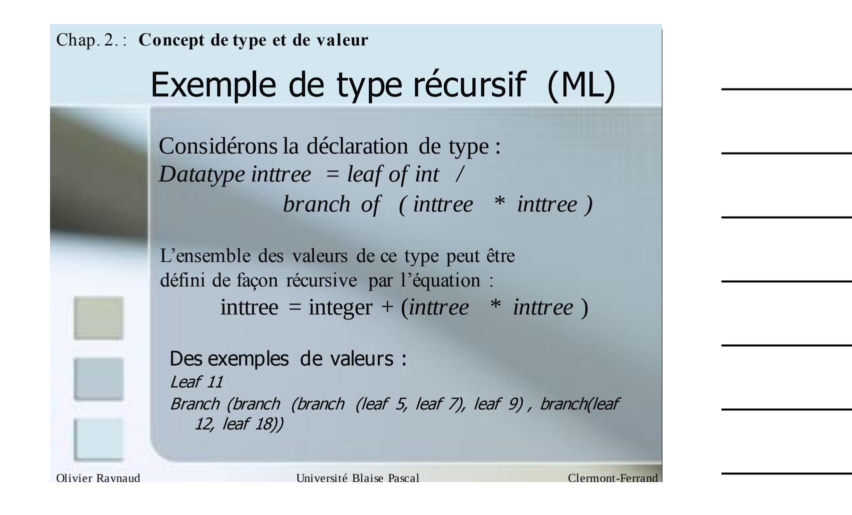

Chap. 2. : Concept de type et de valeur
Les tableaux comme « mapping »
Les tableaux permettent d'implémenter des fonctions dont le domaine de définition est discret et fini.
Conventionnellement, le tableau permet de mettre en correspondance l'ensemble des indices du tableau (ensemble fini, discret, de taille raisonnable) avec l'ensemble des valeurs contenues dans le tableau.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

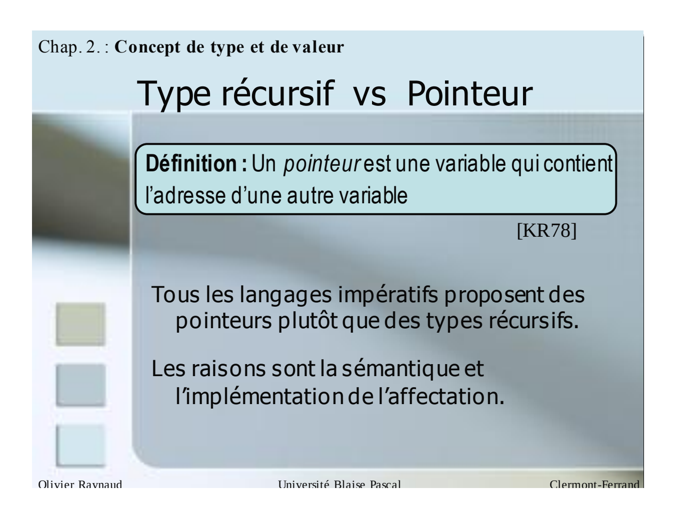

Chap. 2. : Concept de type et de valeur
En Pascal
En pascal la déclaration : array[S] of T défini un mapping de l'ensemble S dans l'ensemble T;
type couleur = (rouge, vert, bleu);
pixel = array [couleur] of 0..1
Une valeur possible du type pixel est :
{rouge → 0; vert → 1, bleu → 0}
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

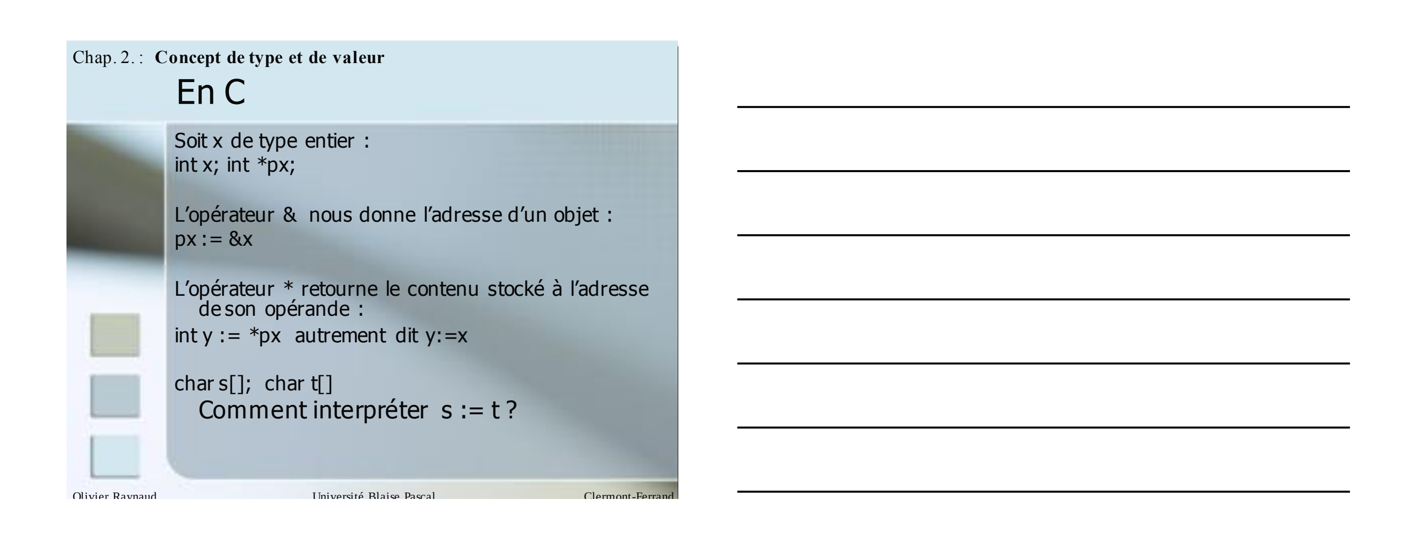

Chap. 2. : Concept de type et de valeur
Les fonctions comme « mapping »
Les fonctions définies par un algorithme, qui à toute valeur de S (l'argument) associe une valeur de T (le résultat), est une implémentation possible de la notion de fonction au sens « mapping ».
La distinction sémantique entre les tableaux et les fonctions est la même qu'entre se souvenir d'une solution ou calculer la solution.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

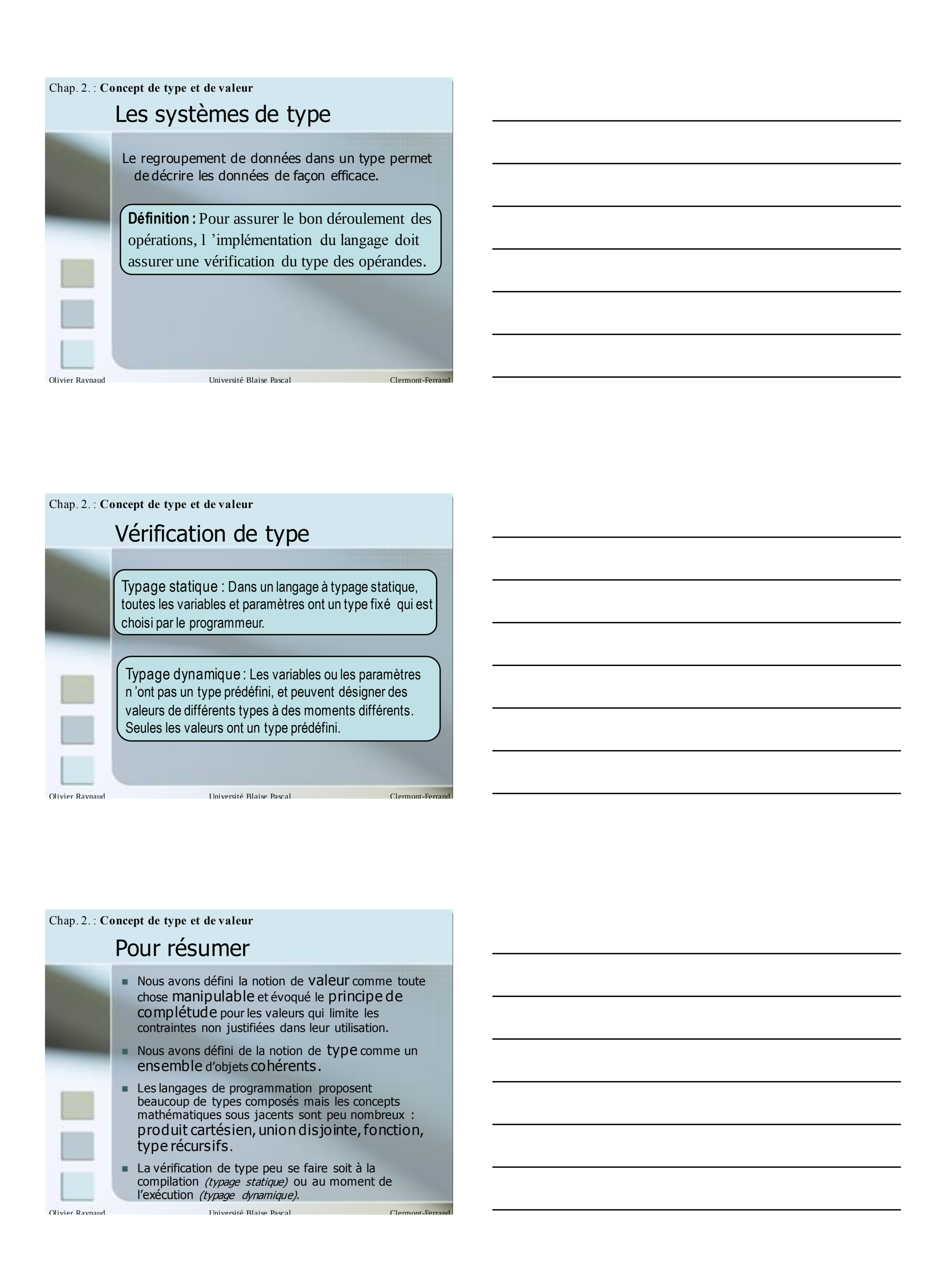

Chap. 2. : Concept de type et de valeur
En Pascal
function pair? (n :integer) : boolean
begin
pair? := (n mod 2 = 0)
end
Cette fonction implémente un mapping de l'ensemble des entiers relatifs dans l'ensemble des valeurs de vérité :
{0 → vrai; +/- 1 → faux, +/- 2 → vrai, +/- 3 → faux}
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

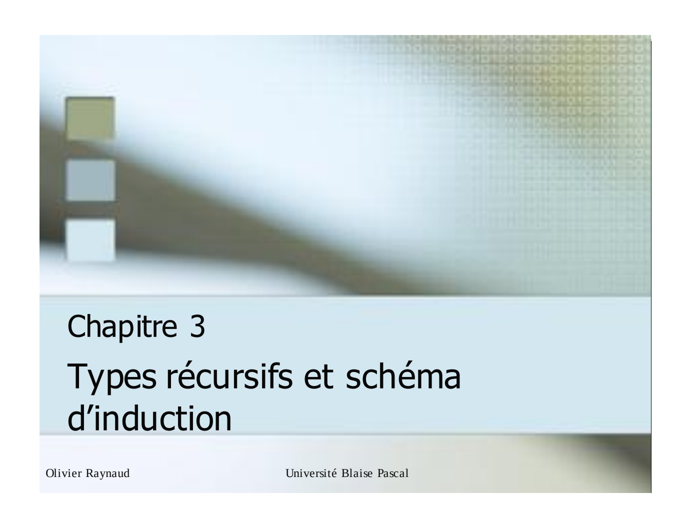

Chap. 2. : Concept de type et de valeur
Les types récursifs
Définition : Un type récursif est un type dont les valeurs sont composées à partir des valeurs du même type.
L'ensemble des valeurs d'un type T récursif peut être défini par une équation de la forme :
T = … T …
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand
Chap. 2. : Concept de type et de valeur
Exemple des listes
Définition : Une liste est une séquence de valeurs de taille quelconque. Son nombre de composants est appelé la longueur de la liste. La seule liste n'ayant aucun composant est appelée la liste vide.
Supposons que l'on désire définir un type L dont les valeurs sont des listes de symboles de l'ensemble S.
L'ensemble des valeurs de ce type peut être défini de façon récursive par l'équation :
L = Unité + (S x L)
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand
Chap. 2. : Concept de type et de valeur
Exemple des strings
Définition : Un string est une séquence de caractères. Ce type de données est disponible dans tous les langages de programmations modernes.
Pourtant aucun consensus n'existe sur la classification des strings.
a) Les strings sont elles des valeurs primitives ou composées?
b) Quels opérations seront disponibles pour le traitement des strings?
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand


Chap. 2. : Concept de type et de valeur
Exemple de type récursif (ML)
Considérons la déclaration de type :
Datatype inttree = leaf of int /
branch of (inttree * inttree)
L'ensemble des valeurs de ce type peut être
défini de façon récursive par l'équation :
inttree = integer + (inttree * inttree)
Des exemples de valeurs :
Leaf 11
Branch (branch (branch (leaf 5, leaf 7), leaf 9) , branch(leaf 12, leaf 18))
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

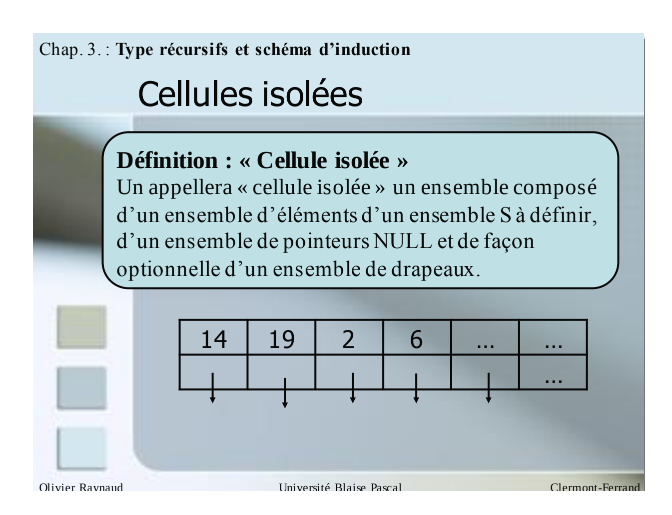

Chap. 2. : Concept de type et de valeur
Type récursif vs Pointeur
Définition : Un pointeur est une variable qui contient l'adresse d'une autre variable
[KR78]
Tous les langages impératifs proposent des pointeurs plutôt que des types récursifs.
Les raisons sont la sémantique et l'implémentation de l'affectation.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand


Chap. 2. : Concept de type et de valeur
En C
Soit x de type entier :
int x; int *px;
L'opérateur & nous donne l'adresse d'un objet :
px := &x
L'opérateur * retourne le contenu stocké à l'adresse de son opérande :
int y := *px autrement dit y:=x
char s[]; char t[]
Comment interpréter s := t ?
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

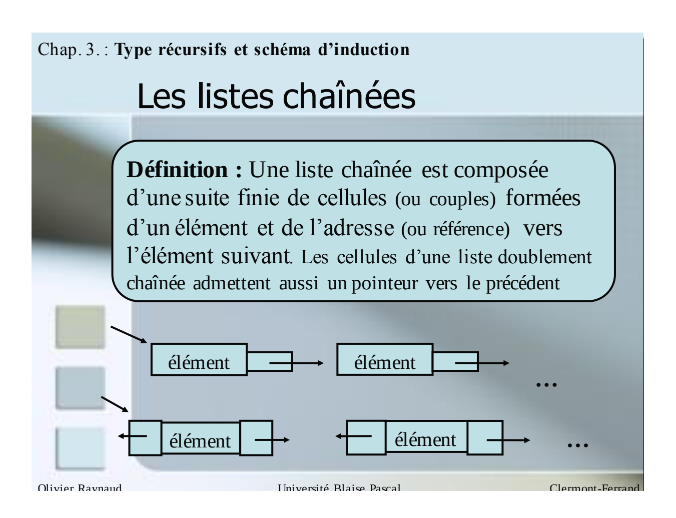

Chap. 2. : Concept de type et de valeur
Les systèmes de type
Le regroupement de données dans un type permet de décrire les données de façon efficace.
Définition : Pour assurer le bon déroulement des opérations, l 'implémentation du langage doit assurer une vérification du type des opérandes.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand
Chap. 2. : Concept de type et de valeur
Vérification de type
Typage statique : Dans un langage à typage statique, toutes les variables et paramètres ont un type fixé qui est choisi par le programmeur.
Typage dynamique : Les variables ou les paramètres n 'ont pas un type prédéfini, et peuvent désigner des valeurs de différents types à des moments différents. Seules les valeurs ont un type prédéfini.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand
Chap. 2. : Concept de type et de valeur
Pour résumer
Nous avons défini la notion de valeur comme toute chose manipulable et évoqué le principe de complétude pour les valeurs qui limite les contraintes non justifiées dans leur utilisation.
Nous avons défini de la notion de type comme un ensemble d'objets cohérents.
Les langages de programmation proposent beaucoup de types composés mais les concepts mathématiques sous jacents sont peu nombreux : produit cartésien, union disjointe, fonction, type récursifs.
La vérification de type peu se faire soit à la compilation (typage statique) ou au moment de l'exécution (typage dynamique).
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand



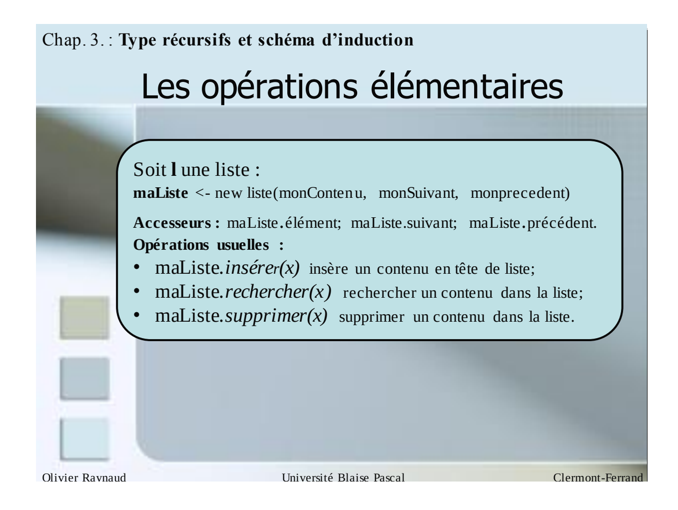
Chapitre 3
Types récursifs et schéma d'induction
Olivier Raynaud
Université Blaise Pascal



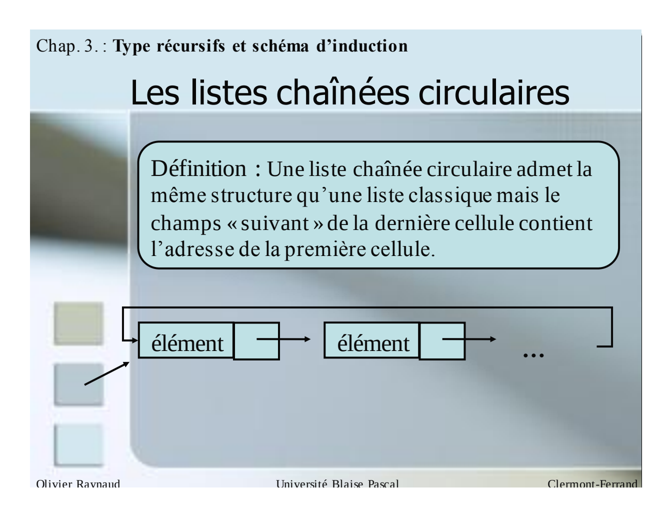
Chap. 3. : Type récursifs et schéma d'induction
Cellules isolées
Définition : « Cellule isolée »
Un appellera « cellule isolée » un ensemble composé d'un ensemble d'éléments d'un ensemble S à définir, d'un ensemble de pointeurs NULL et de façon optionnelle d'un ensemble de drapeaux.
14
19
2
6
…
…
…
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand



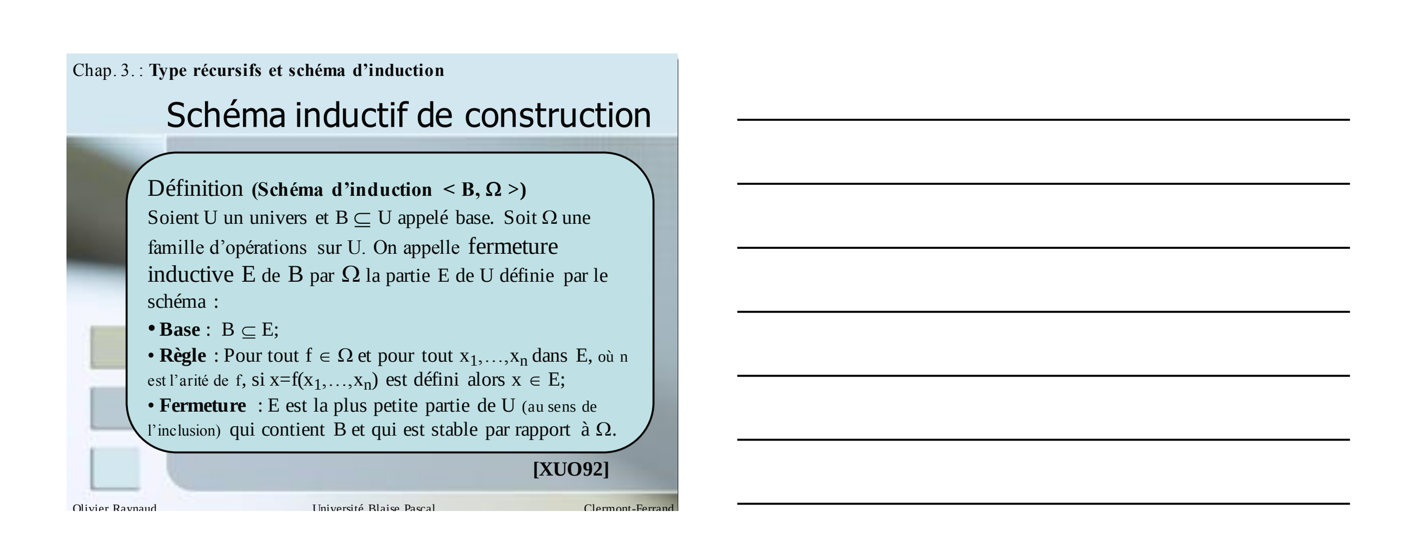
Chap. 3. : Type récursifs et schéma d'induction
Les listes chaînées
Définition : Une liste chaînée est composée d'une suite finie de cellules (ou couples) formées d'un élément et de l'adresse (ou référence) vers l'élément suivant. Les cellules d'une liste doublement chaînée admettent aussi un pointeur vers le précédent
élément
élément
…
élément
élément
…
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand



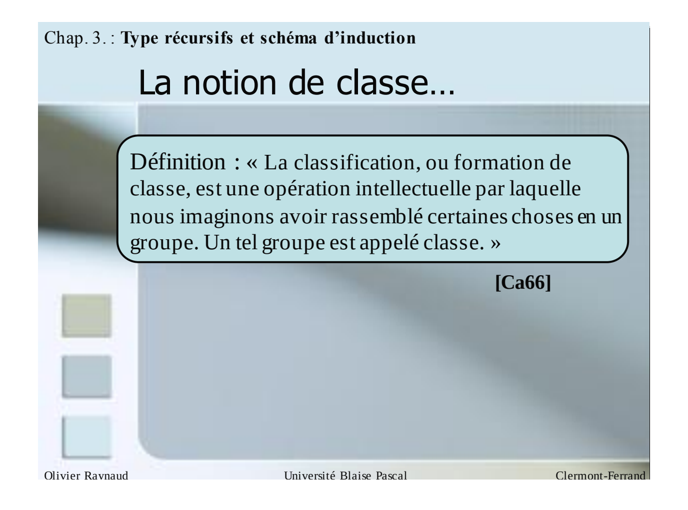
Chap. 3. : Type récursifs et schéma d'induction
Les opérations élémentaires
Soit l une liste :
maListe <- new liste(monContenu, monSuivant, monprecedent)
Accesseurs : maListe.élément; maListe.suivant; maListe.précédent.
Opérations usuelles :
• maListe.insérer(x) insère un contenu en tête de liste;
• maListe.rechercher(x) rechercher un contenu dans la liste;
• maListe.supprimer(x) supprimer un contenu dans la liste.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

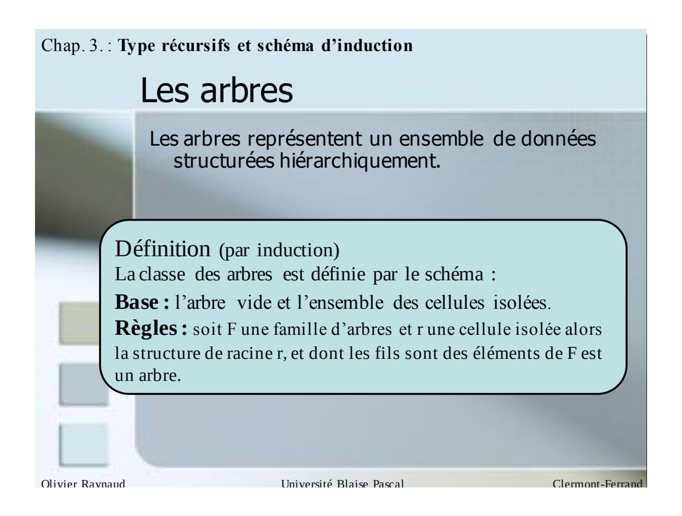


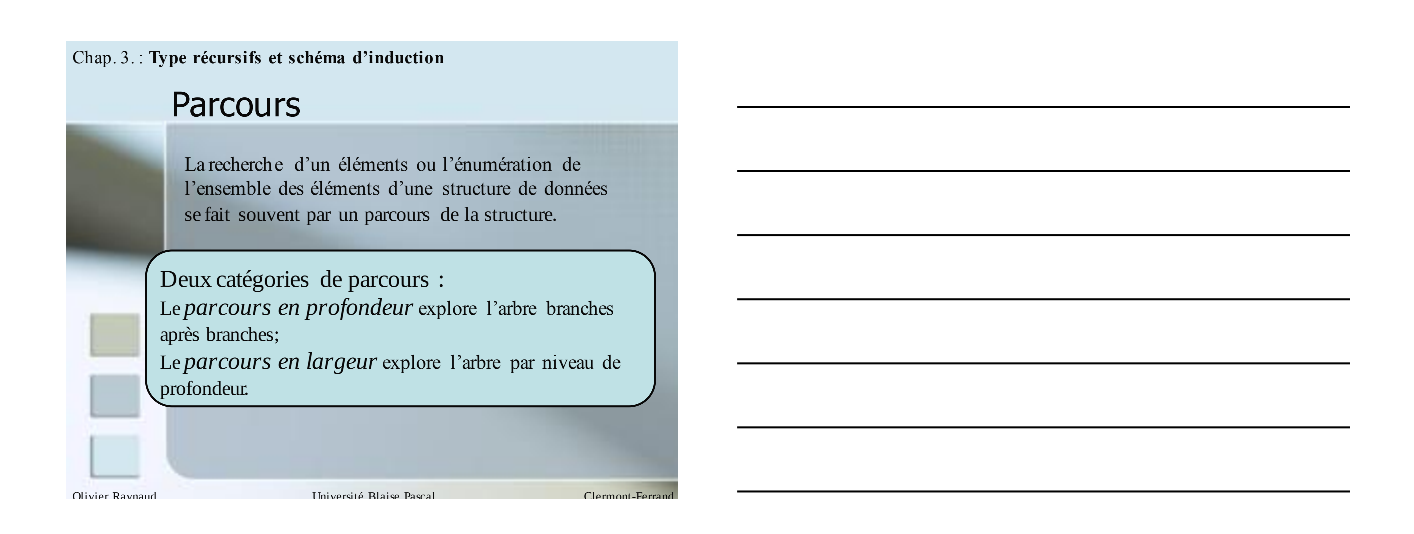
Chap. 3. : Type récursifs et schéma d'induction
Les listes chaînées circulaires
Définition : Une liste chaînée circulaire admet la même structure qu'une liste classique mais le champs « suivant » de la dernière cellule contient l'adresse de la première cellule.
élément
élément
...
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

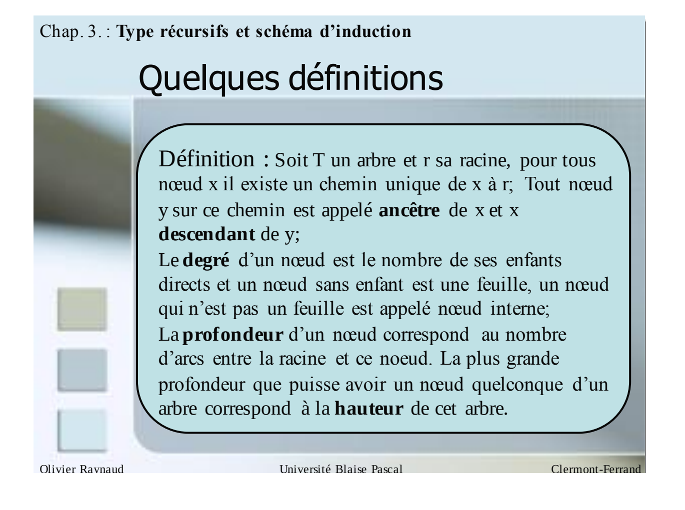


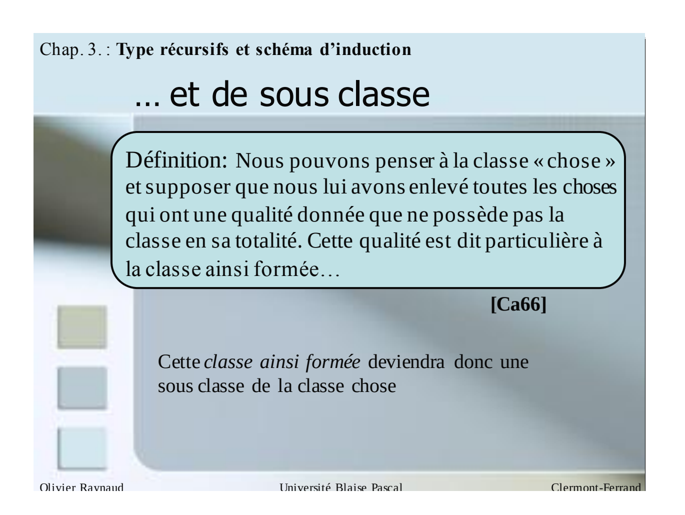
Chap. 3. : Type récursifs et schéma d'induction
Schéma inductif de construction
Définition (Schéma d'induction < B, Ω >)
Soient U un univers et B ⊆ U appelé base. Soit Ω une famille d'opérations sur U. On appelle fermeture inductive E de B par Ω la partie E de U définie par le schéma :
• Base : B ⊆ E;
• Règle : Pour tout f ∈ Ω et pour tout x1,…,xn dans E, où n est l'arité de f, si x=f(x1,…,xn) est défini alors x ∈ E;
• Fermeture : E est la plus petite partie de U (au sens de l'inclusion) qui contient B et qui est stable par rapport à Ω.
[XUO92]
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

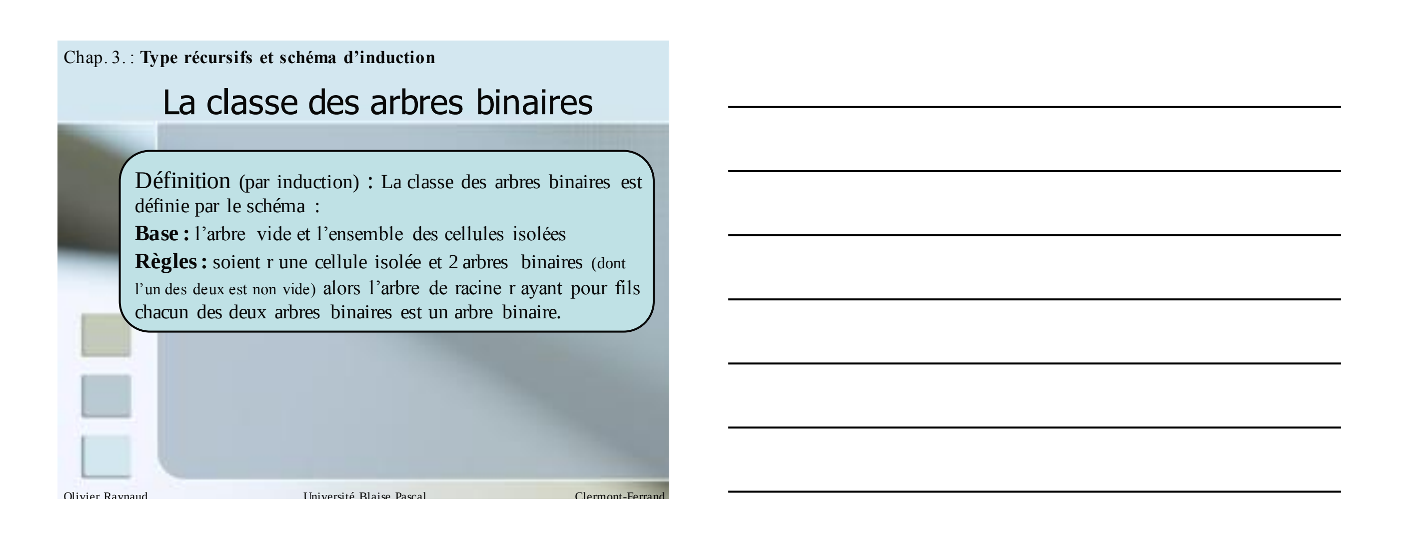

Chap. 3. : Type récursifs et schéma d'induction
La notion de classe…
Définition : « La classification, ou formation de classe, est une opération intellectuelle par laquelle nous imaginons avoir rassemblé certaines choses en un groupe. Un tel groupe est appelé classe. »
[Ca66]
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

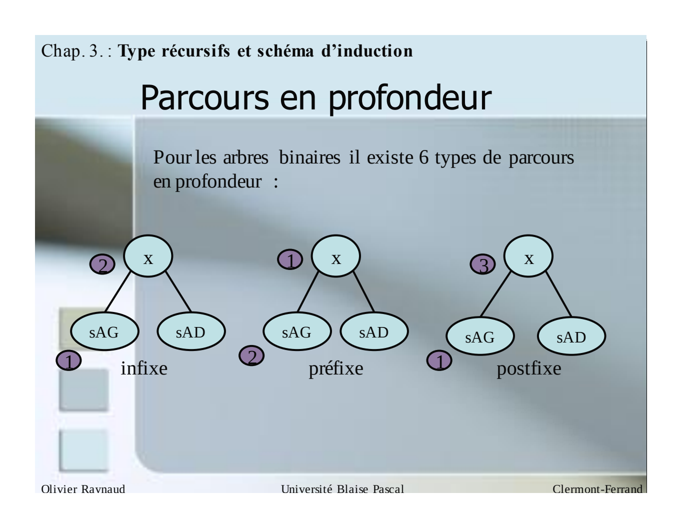

Chap. 3. : Type récursifs et schéma d'induction
Les arbres
Les arbres représentent un ensemble de données structurées hiérarchiquement.
Définition (par induction)
La classe des arbres est définie par le schéma :
Base : l'arbre vide et l'ensemble des cellules isolées.
Règles : soit F une famille d'arbres et r une cellule isolée alors la structure de racine r, et dont les fils sont des éléments de F est un arbre.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

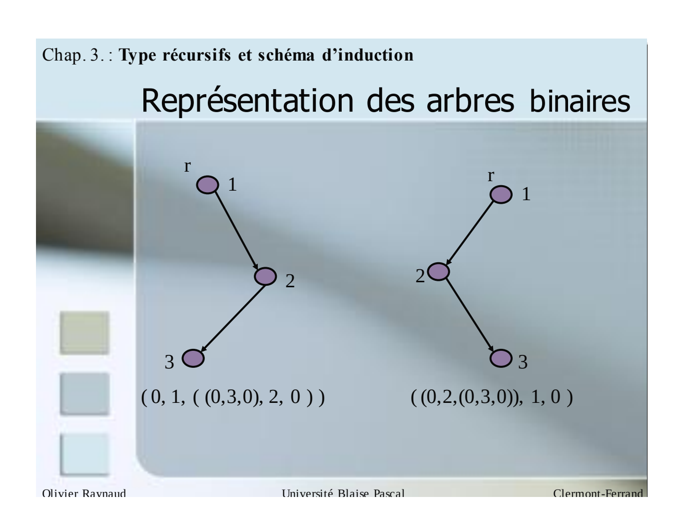

Chap. 3. : Type récursifs et schéma d'induction
Parcours
La recherche d'un éléments ou l'énumération de l'ensemble des éléments d'une structure de données se fait souvent par un parcours de la structure.
Deux catégories de parcours :
Le parcours en profondeur explore l'arbre branches après branches;
Le parcours en largeur explore l'arbre par niveau de profondeur.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

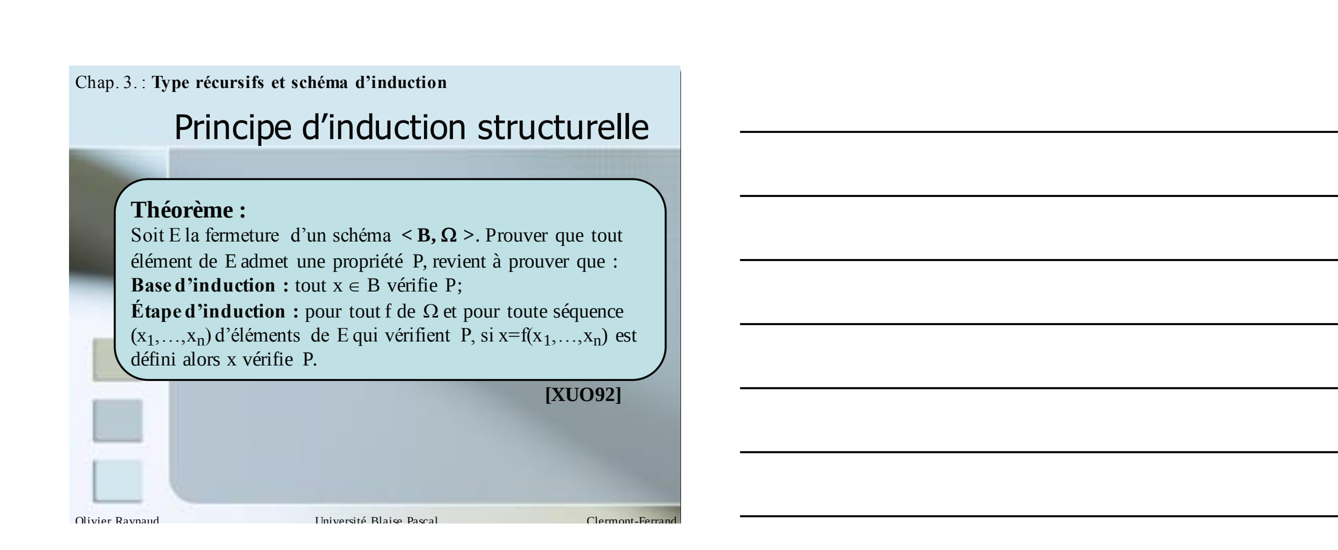

Chap. 3. : Type récursifs et schéma d'induction
Quelques définitions
Définition : Soit T un arbre et r sa racine, pour tous nœud x il existe un chemin unique de x à r; Tout nœud y sur ce chemin est appelé ancêtre de x et x descendant de y;
Le degré d'un nœud est le nombre de ses enfants directs et un nœud sans enfant est une feuille, un nœud qui n'est pas un feuille est appelé nœud interne;
La profondeur d'un nœud correspond au nombre d'arcs entre la racine et ce noeud. La plus grande profondeur que puisse avoir un nœud quelconque d'un arbre correspond à la hauteur de cet arbre.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

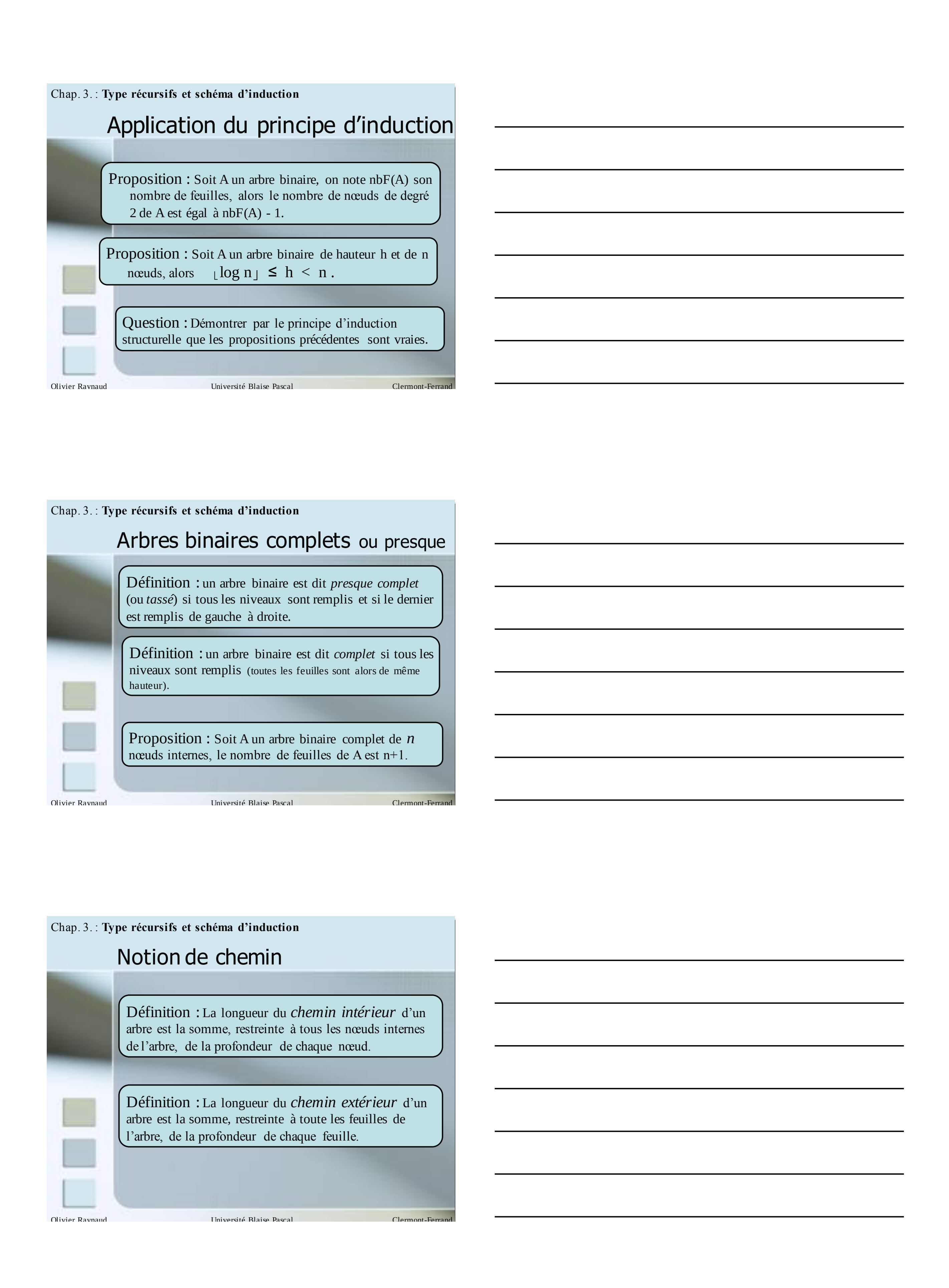

Chap. 3. : Type récursifs et schéma d'induction
… et de sous classe
Définition: Nous pouvons penser à la classe « chose » et supposer que nous lui avons enlevé toutes les choses qui ont une qualité donnée que ne possède pas la classe en sa totalité. Cette qualité est dit particulière à la classe ainsi formée…
[Ca66]
Cette classe ainsi formée deviendra donc une sous classe de la classe chose
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

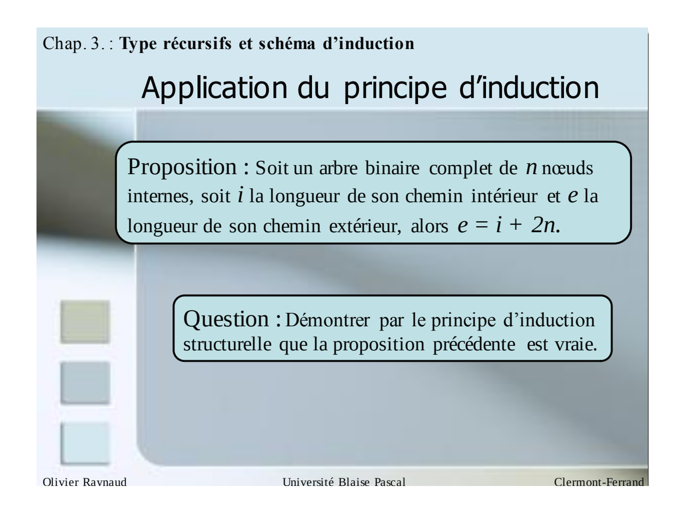

Chap. 3. : Type récursifs et schéma d'induction
La classe des arbres binaires
Définition (par induction) : La classe des arbres binaires est définie par le schéma :
Base : l'arbre vide et l'ensemble des cellules isolées
Règles : soient r une cellule isolée et 2 arbres binaires (dont l'un des deux est non vide) alors l'arbre de racine r ayant pour fils chacun des deux arbres binaires est un arbre binaire.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

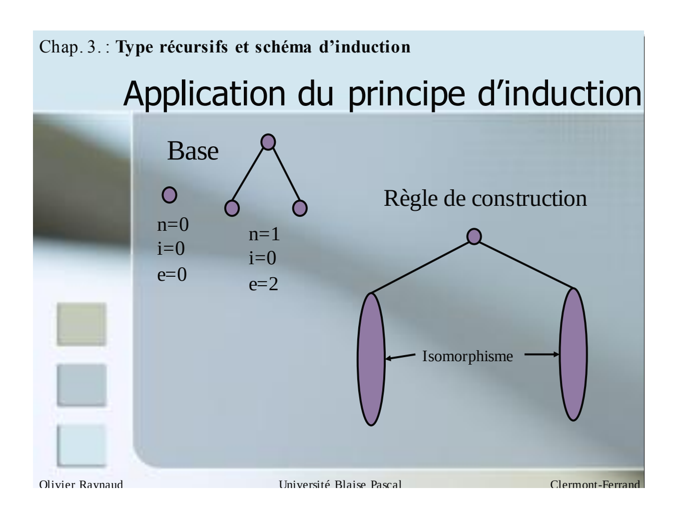

Chap. 3. : Type récursifs et schéma d'induction
Parcours en profondeur
Pour les arbres binaires il existe 6 types de parcours en profondeur :
x
sAG
sAD
2
1
infixe
1
2
préfixe
3
1
postfixe
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

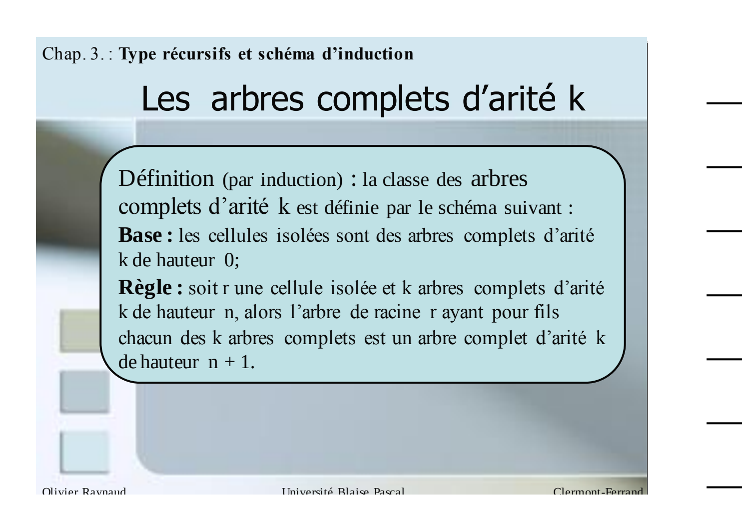

Chap. 3. : Type récursifs et schéma d'induction
Représentation des arbres binaires
r
1
2
3
(0, 1, ((0,3,0), 2, 0))
((0,2,(0,3,0)), 1, 0)
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

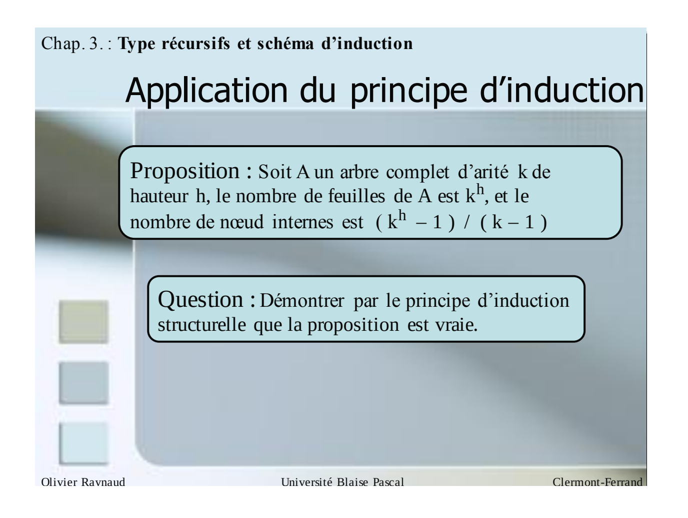

Chap. 3. : Type récursifs et schéma d'induction
Principe d'induction structurelle
Théorème :
Soit E la fermeture d'un schéma < B, Ω >. Prouver que tout élément de E admet une propriété P, revient à prouver que :
Base d'induction : tout x ∈ B vérifie P;
Étape d'induction : pour tout f de Ω et pour toute séquence (x_1,…,x_n) d'éléments de E qui vérifient P, si x=f(x_1,…,x_n) est défini alors x vérifie P.
[XUO92]
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

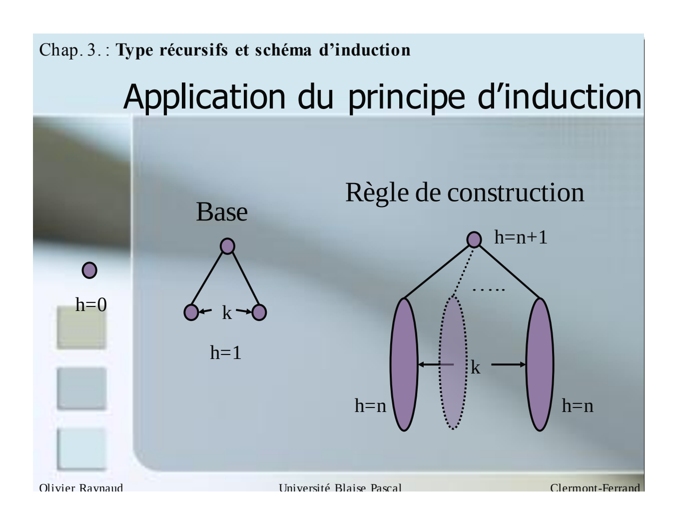

Chap. 3. : Type récursifs et schéma d'induction
Application du principe d'induction
Proposition : Soit A un arbre binaire, on note nbF(A) son nombre de feuilles, alors le nombre de nœuds de degré 2 de A est égal à nbF(A) - 1.
Proposition : Soit A un arbre binaire de hauteur h et de n nœuds, alors ⌊log n⌋ ≤ h < n .
Question : Démontrer par le principe d'induction structurelle que les propositions précédentes sont vraies.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand
Chap. 3. : Type récursifs et schéma d'induction
Arbres binaires complets ou presque
Définition : un arbre binaire est dit presque complet (ou tassé) si tous les niveaux sont remplis et si le dernier est remplis de gauche à droite.
Définition : un arbre binaire est dit complet si tous les niveaux sont remplis (toutes les feuilles sont alors de même hauteur).
Proposition : Soit A un arbre binaire complet de n nœuds internes, le nombre de feuilles de A est n+1.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand
Chap. 3. : Type récursifs et schéma d'induction
Notion de chemin
Définition : La longueur du chemin intérieur d'un arbre est la somme, restreinte à tous les nœuds internes de l'arbre, de la profondeur de chaque nœud.
Définition : La longueur du chemin extérieur d'un arbre est la somme, restreinte à toute les feuilles de l'arbre, de la profondeur de chaque feuille.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

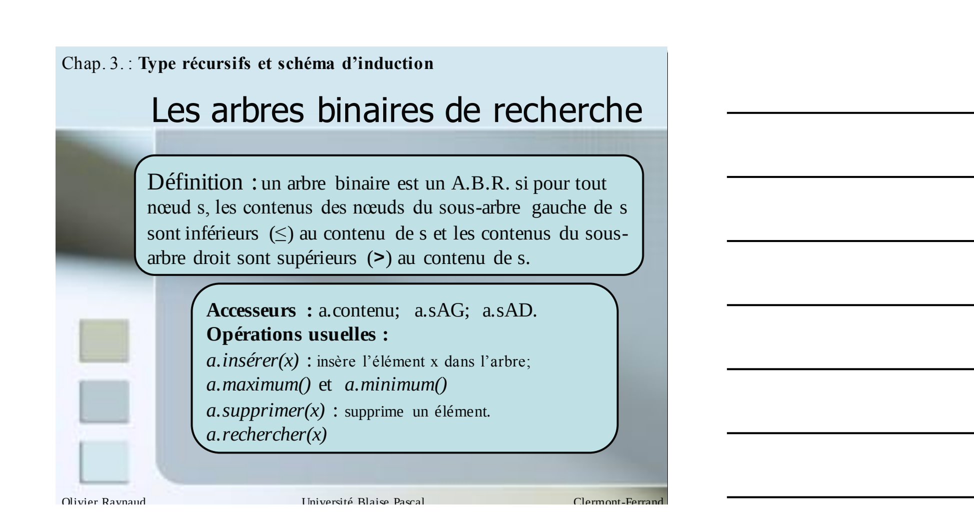

Chap. 3. : Type récursifs et schéma d'induction
Application du principe d'induction
Proposition : Soit un arbre binaire complet de n nœuds internes, soit i la longueur de son chemin intérieur et e la longueur de son chemin extérieur, alors e = i + 2n.
Question : Démontrer par le principe d'induction structurelle que la proposition précédente est vraie.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

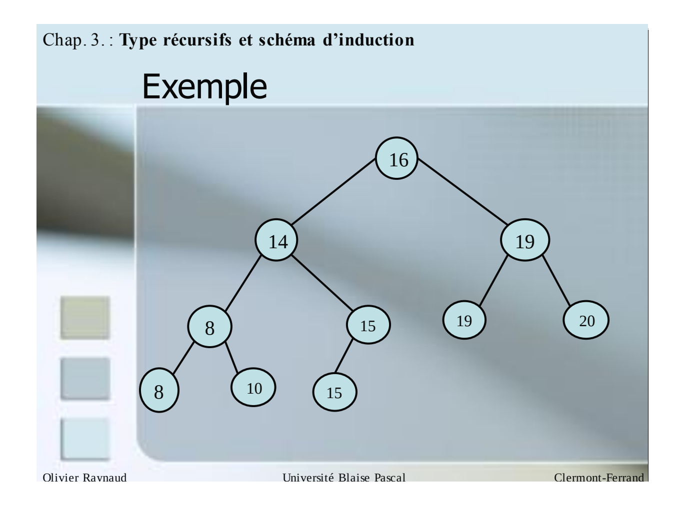

Chap. 3. : Type récursifs et schéma d'induction
Application du principe d'induction
Base
n=0
i=0
e=0
n=1
i=0
e=2
Règle de construction
Isomorphisme
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

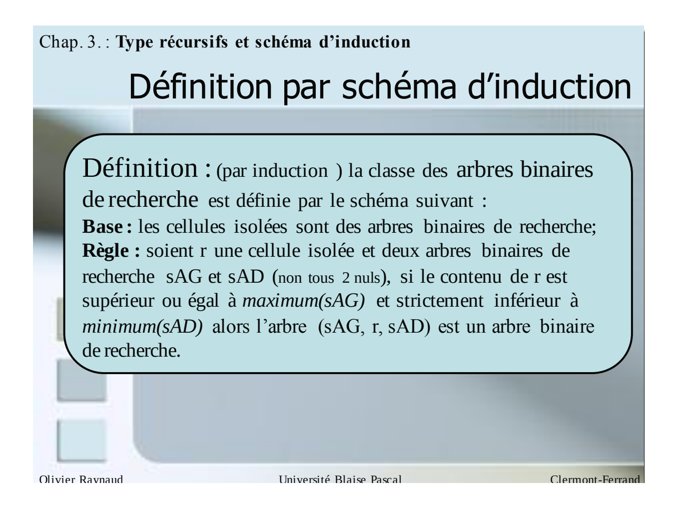

Chap. 3. : Type récursifs et schéma d'induction
Les arbres complets d'arité k
Définition (par induction) : la classe des arbres complets d'arité k est définie par le schéma suivant :
Base : les cellules isolées sont des arbres complets d'arité k de hauteur 0;
Règle : soit r une cellule isolée et k arbres complets d'arité k de hauteur n, alors l'arbre de racine r ayant pour fils chacun des k arbres complets est un arbre complet d'arité k de hauteur n + 1.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

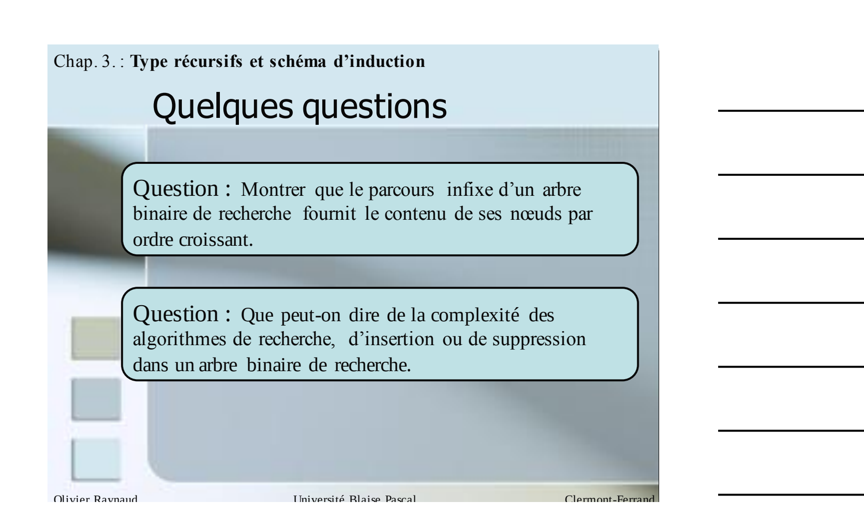

Chap. 3. : Type récursifs et schéma d'induction
Application du principe d'induction
Proposition : Soit A un arbre complet d'arité k de hauteur h, le nombre de feuilles de A est k^h, et le nombre de nœud internes est $(k^h - 1) / (k - 1)$
Question : Démontrer par le principe d'induction structurelle que la proposition est vraie.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

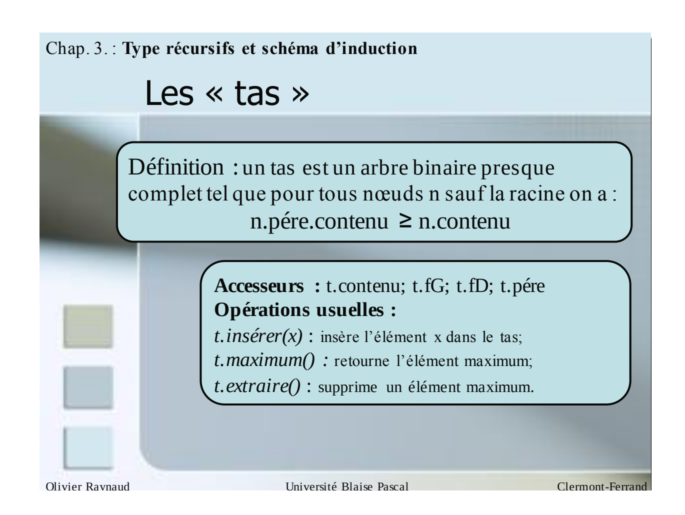

Chap. 3. : Type récursifs et schéma d'induction
Application du principe d'induction
Base
Règle de construction
h=0
k
h=1
h=n+1
…..
k
h=n
h=n
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

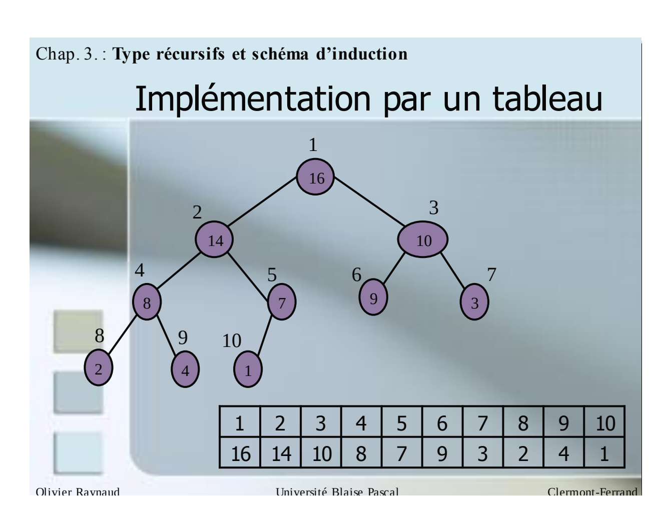

Chap. 3. : Type récursifs et schéma d'induction
Les arbres binaires de recherche
Définition : un arbre binaire est un A.B.R. si pour tout nœud s, les contenus des nœuds du sous-arbre gauche de s sont inférieurs (≤) au contenu de s et les contenus du sous-arbre droit sont supérieurs (>) au contenu de s.
Accesseurs : a.contenu; a.sAG; a.sAD.
Opérations usuelles :
a.insérer(x) : insère l'élément x dans l'arbre;
a.maximum() et a.minimum()
a.supprimer(x) : supprime un élément.
a.rechercher(x)
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand


Chap. 3. : Type récursifs et schéma d'induction
Exemple
16
14
19
8
15
19
20
8
10
15
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

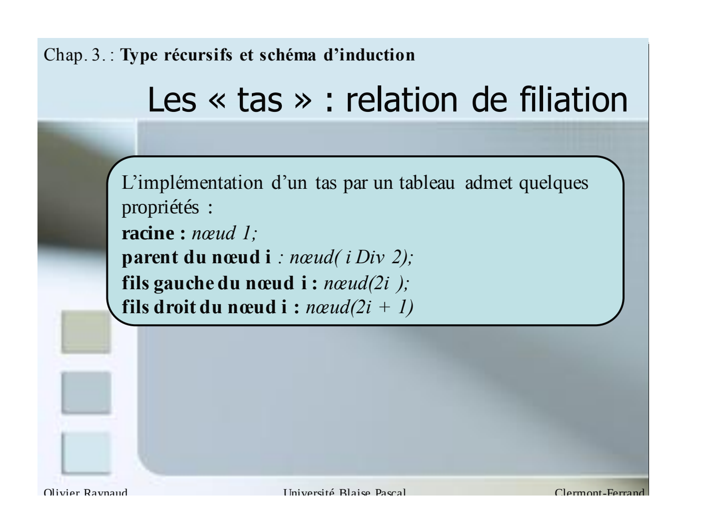

Chap. 3. : Type récursifs et schéma d'induction
Définition par schéma d'induction
Définition : (par induction) la classe des arbres binaires de recherche est définie par le schéma suivant :
Base : les cellules isolées sont des arbres binaires de recherche;
Règle : soient r une cellule isolée et deux arbres binaires de recherche sAG et sAD (non tous 2 nuls), si le contenu de r est supérieur ou égal à maximum(sAG) et strictement inférieur à minimum(sAD) alors l'arbre (sAG, r, sAD) est un arbre binaire de recherche.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand


Chap. 3. : Type récursifs et schéma d'induction
Quelques questions
Question : Montrer que le parcours infixe d'un arbre binaire de recherche fournit le contenu de ses nœuds par ordre croissant.
Question : Que peut-on dire de la complexité des algorithmes de recherche, d'insertion ou de suppression dans un arbre binaire de recherche.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

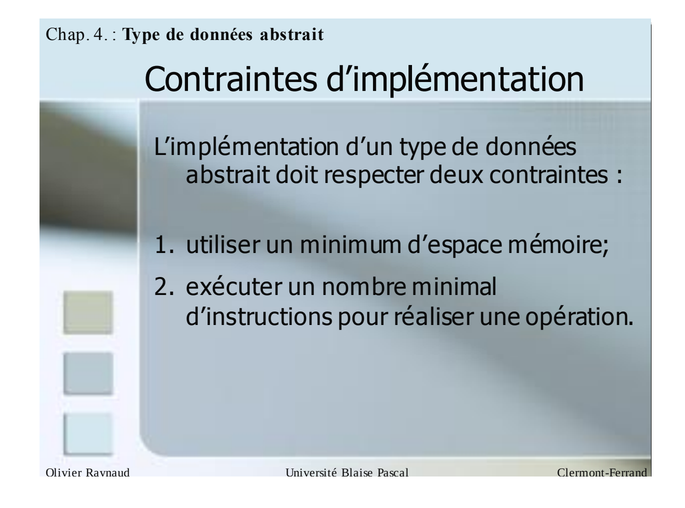


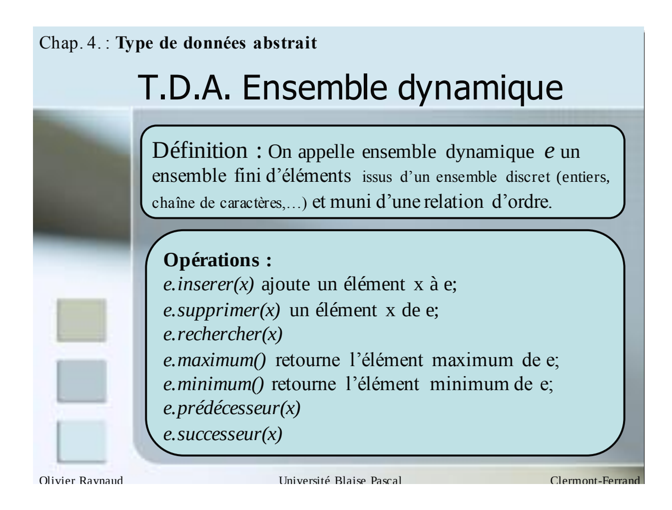


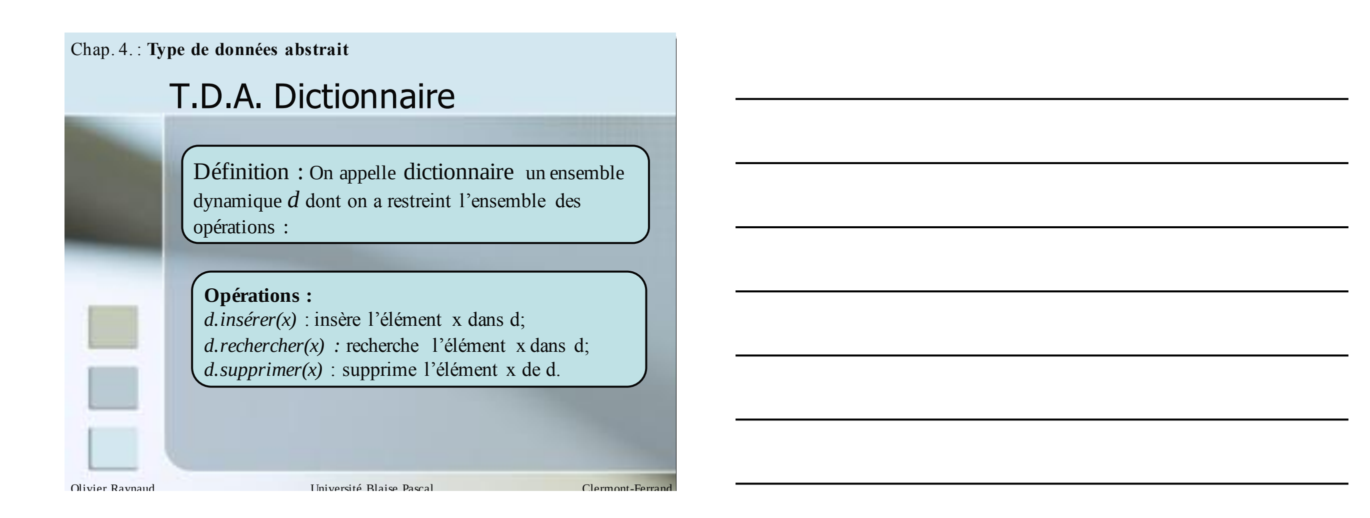
1	2	3	4	5	6	7	8	9	10
16	14	10	8	7	9	3	2	4	1




Chap. 3. : Type récursifs et schéma d'induction
Pour résumer
Nous avons défini la notion de cellule (des contenus, des pointeurs, des drapeaux). Nous avons alors montré comment construire des listes chaînées, simple, double, circulaire.
Nous avons montré le lien entre définition récursive et définition par construction inductive (une base, des règles de production).
Nous avons montré comment démontrer des propriétés sur des ensembles d'objets potentiellement infinis grâce au principe d'induction structurelle.
Nous avons donné les spécifications des structures d'aBR et des tas.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand
Chapitre 4
Les types de données abstraits
Olivier Raynaud
Université Blaise Pascal
Chap. 4. : Type de données abstrait
Définition
Un type de données abstrait est composé d'un ensemble d'objets, similaires dans la forme et dans le comportement, et d'un ensemble d'opérations sur ces objets.
L'implémentation d'un T.D.A. ne suis pas de schéma préétabli. Il dépend des objets manipulés et des opérations disponibles pour leur manipulation.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

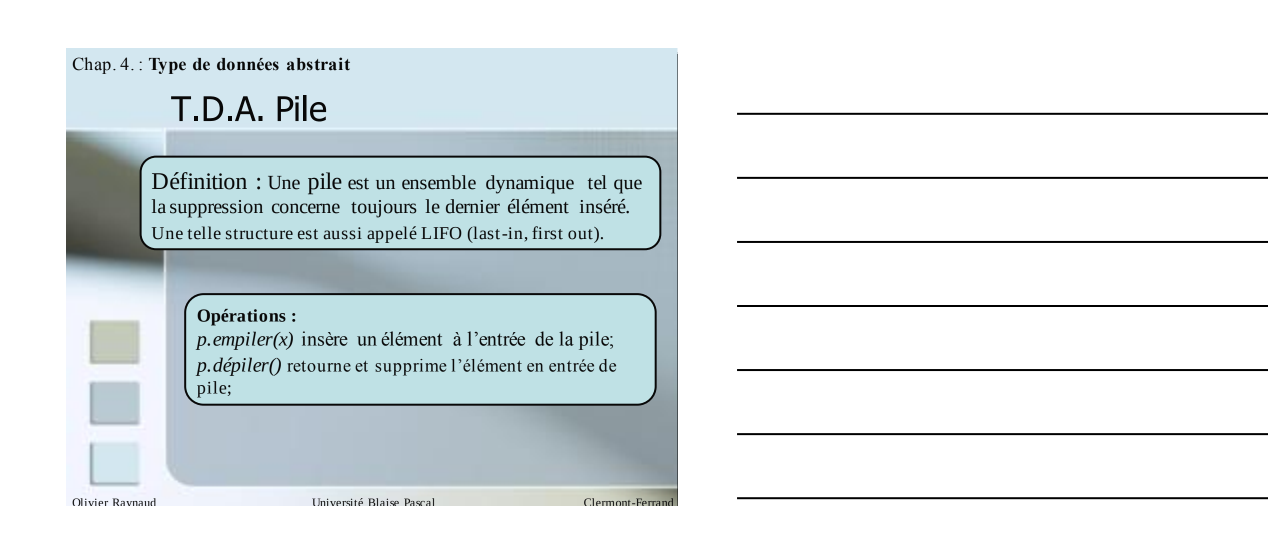

Chap. 4. : Type de données abstrait
Contraintes d’implémentation
L’implémentation d’un type de données abstrait doit respecter deux contraintes :
1. utiliser un minimum d’espace mémoire;
2. exécuter un nombre minimal d’instructions pour réaliser une opération.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

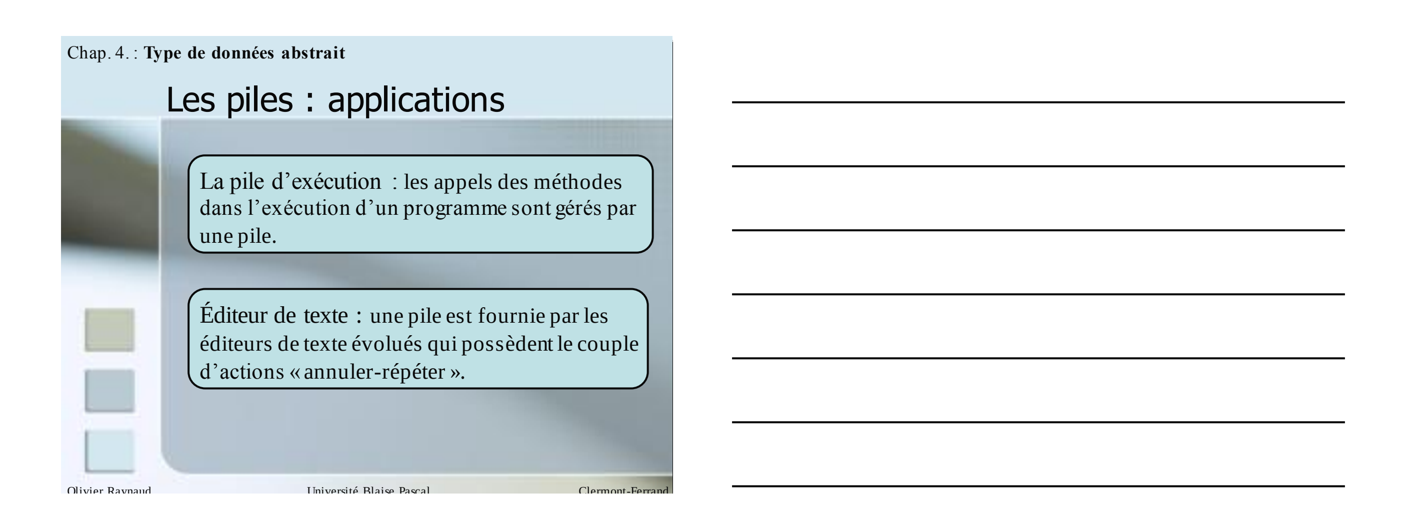

Chap. 4. : Type de données abstrait
T.D.A. Ensemble dynamique
Définition : On appelle ensemble dynamique e un ensemble fini d’éléments issus d’un ensemble discret (entiers, chaîne de caractères,…) et muni d’une relation d’ordre.
Opérations :
e.inserer(x) ajoute un élément x à e;
e.supprimer(x) un élément x de e;
e.rechercher(x)
e.maximum() retourne l’élément maximum de e;
e.minimum() retourne l’élément minimum de e;
e.prédécesseur(x)
e.successeur(x)
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

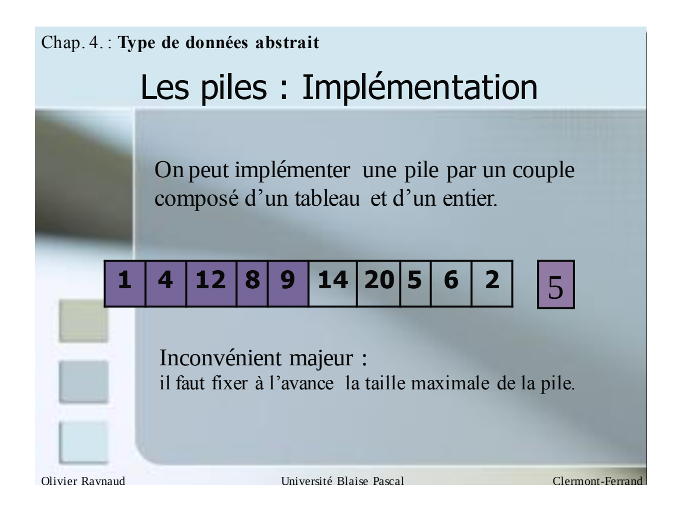

Chap. 4. : Type de données abstrait
T.D.A. Dictionnaire
Définition : On appelle dictionnaire un ensemble dynamique d dont on a restreint l’ensemble des opérations :
Opérations :
d.insérer(x) : insère l’élément x dans d;
d.rechercher(x) : recherche l’élément x dans d;
d.supprimer(x) : supprime l’élément x de d.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

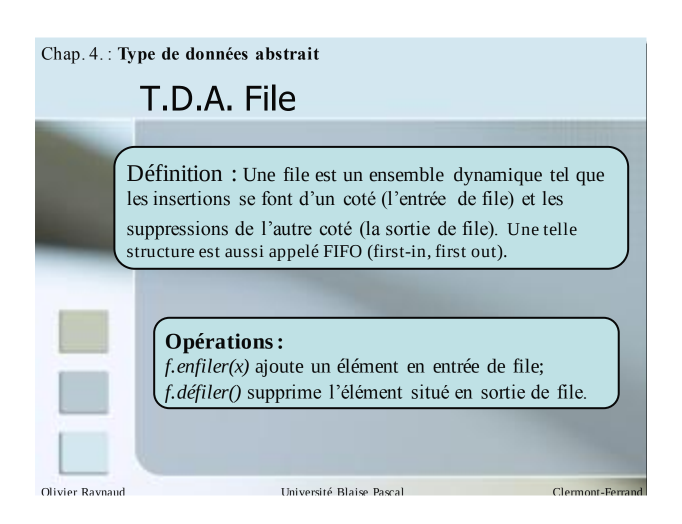
Dictionnaire : Implémentation

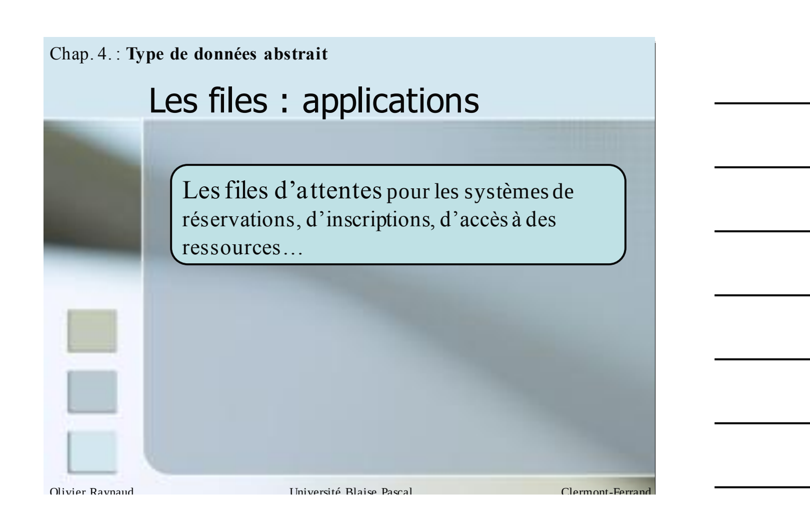
Structure de données	Rechercher	Insérer	Supprimer
Tableau non ordonné	O(n)	O(1)	O(1)
Liste non ordonnée	O(n)	O(1)	O(1)
Tableau ordonné	O(log n)	O(n)	O(n)
Liste ordonnée	O(n)	O(1)	O(1)
Arbre de recherche	O(h)	O(h)	O(h)
Tas	O(n)	O(h)	O(h)

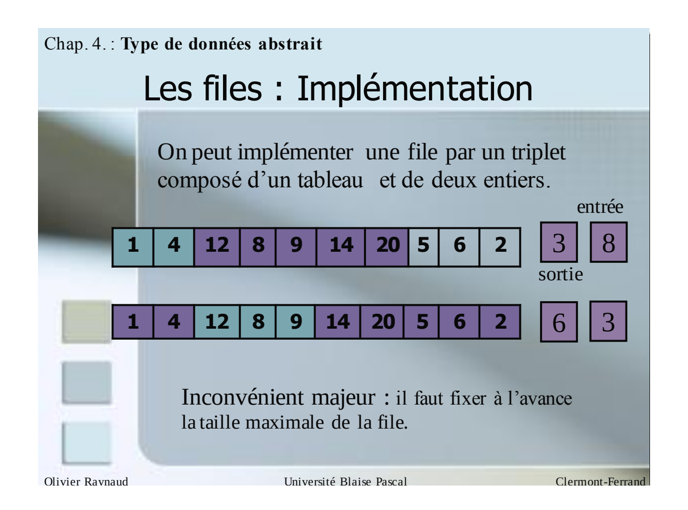


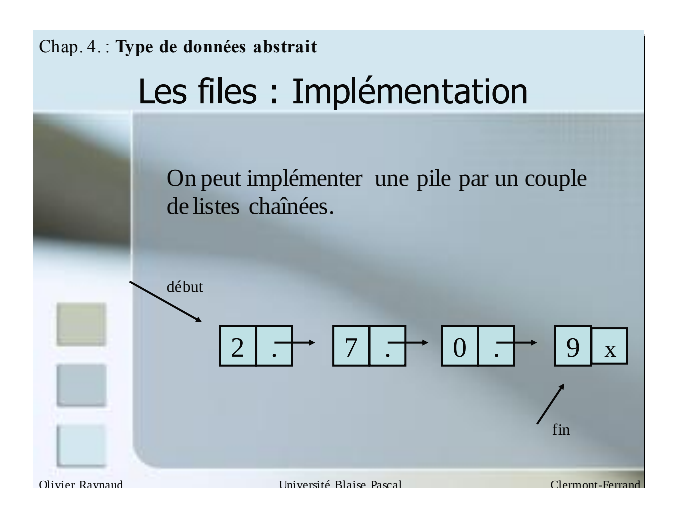


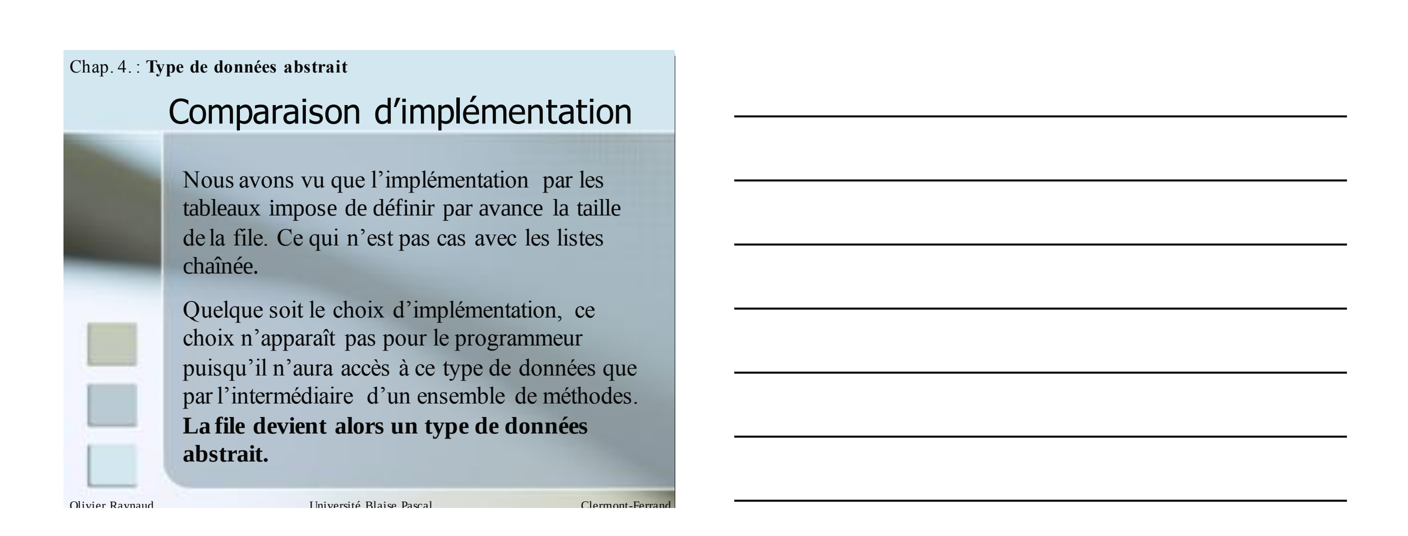

Chap. 4. : Type de données abstrait
Les piles : Implémentation
On peut implémenter une pile par un couple composé d'un tableau et d'un entier.
1 4 12 8 9 14 20 5 6 2
5
Inconvénient majeur :
il faut fixer à l'avance la taille maximale de la pile.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

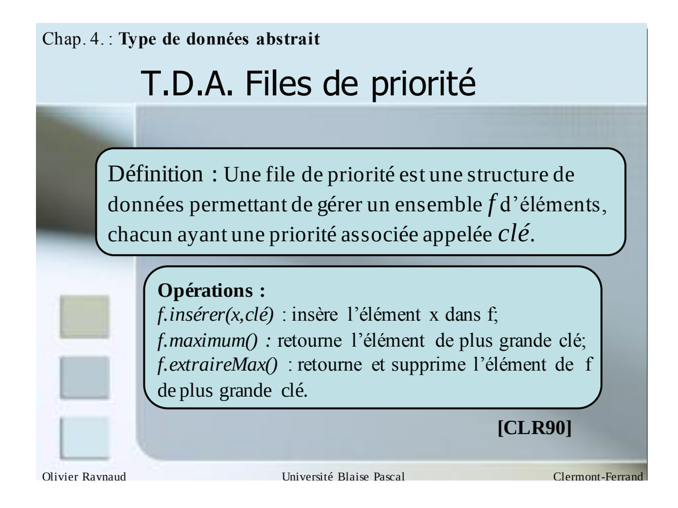

Chap. 4. : Type de données abstrait
T.D.A. File
Définition : Une file est un ensemble dynamique tel que les insertions se font d'un coté (l'entrée de file) et les suppressions de l'autre coté (la sortie de file). Une telle structure est aussi appelé FIFO (first-in, first out).
Opérations :
f.enfiler(x) ajoute un élément en entrée de file;
f.défiler() supprime l'élément situé en sortie de file.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

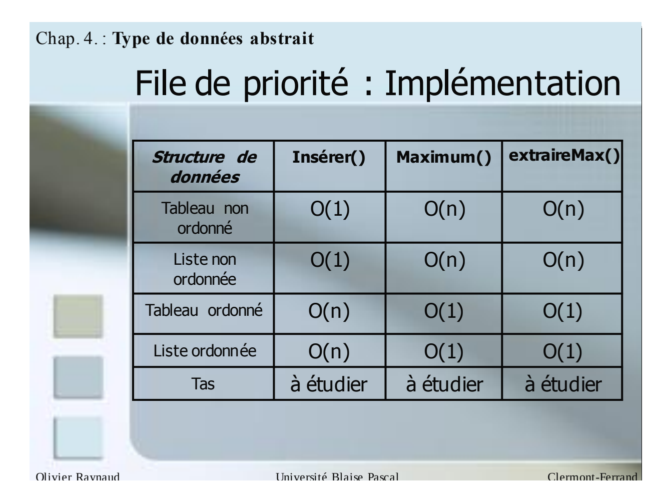

Chap. 4. : Type de données abstrait
Les files : applications
Les files d'attentes pour les systèmes de réservations, d'inscriptions, d'accès à des ressources…
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand


Chap. 4. : Type de données abstrait
Les files : Implémentation
On peut implémenter une file par un triplet composé d'un tableau et de deux entiers.
entrée
1 4 12 8 9 14 20 5 6 2
3 8
sortie
1 4 12 8 9 14 20 5 6 2
6 3
Inconvénient majeur : il faut fixer à l'avance la taille maximale de la file.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

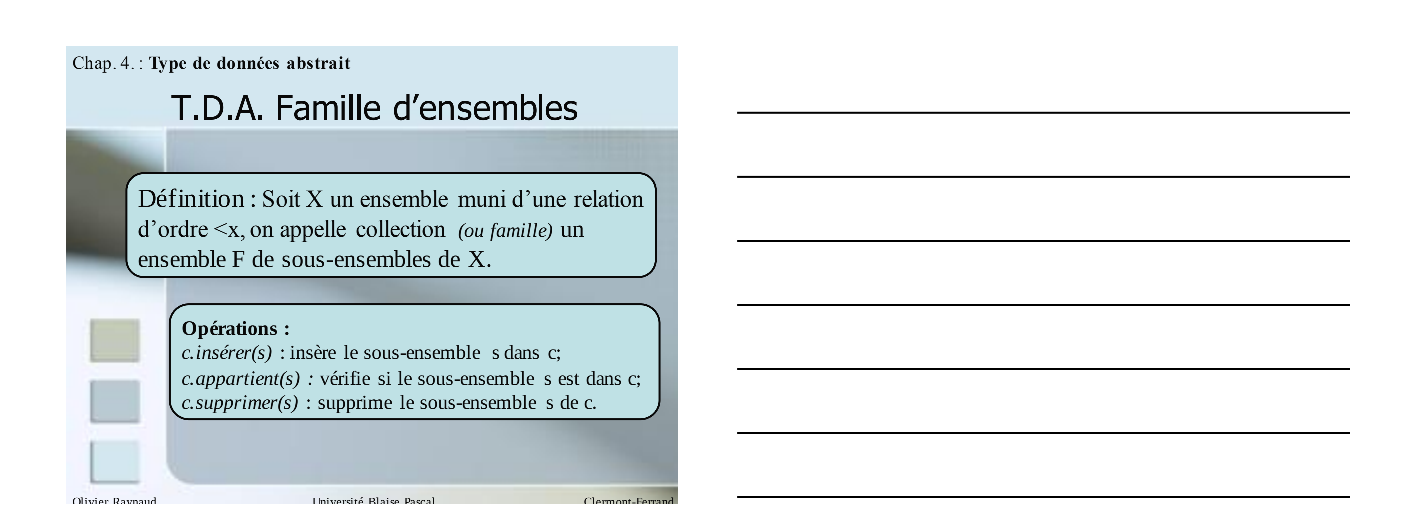

Chap. 4. : Type de données abstrait
Les files : Implémentation
On peut implémenter une pile par un couple de listes chaînées.
début
2 .
7 .
0 .
9 x
fin
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

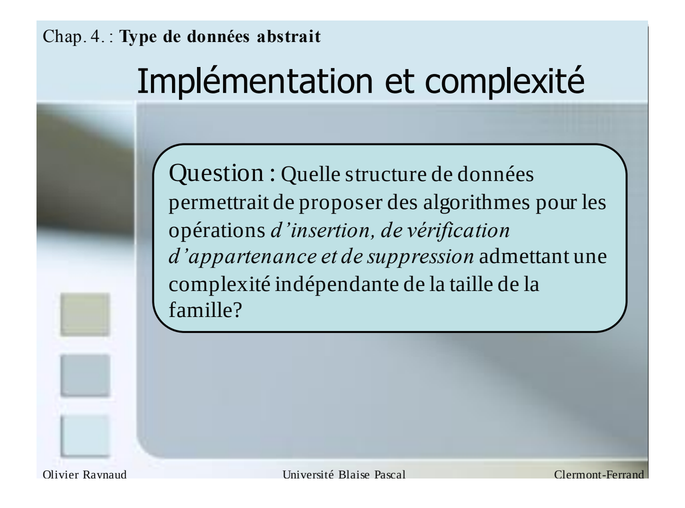

Chap. 4. : Type de données abstrait
Comparaison d'implémentation
Nous avons vu que l'implémentation par les tableaux impose de définir par avance la taille de la file. Ce qui n'est pas cas avec les listes chaînée.
Quelque soit le choix d'implémentation, ce choix n'apparaît pas pour le programmeur puisqu'il n'aura accès à ce type de données que par l'intermédiaire d'un ensemble de méthodes.
La file devient alors un type de données abstrait.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

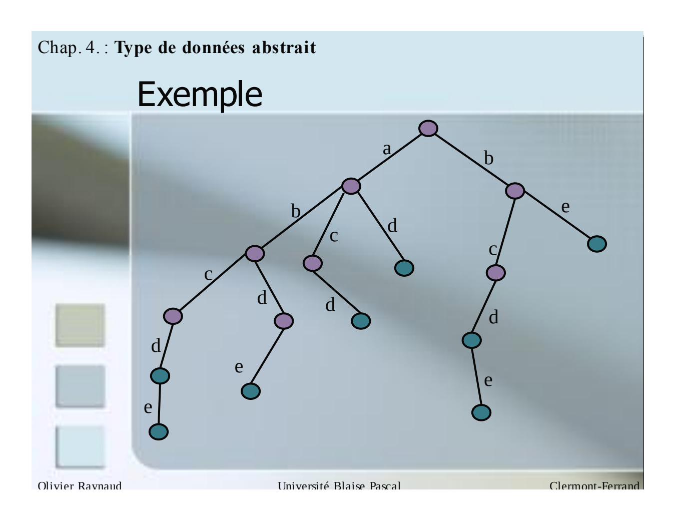


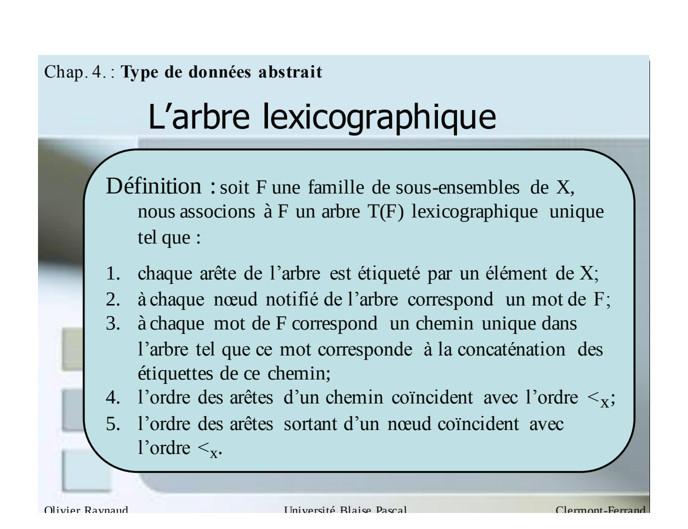


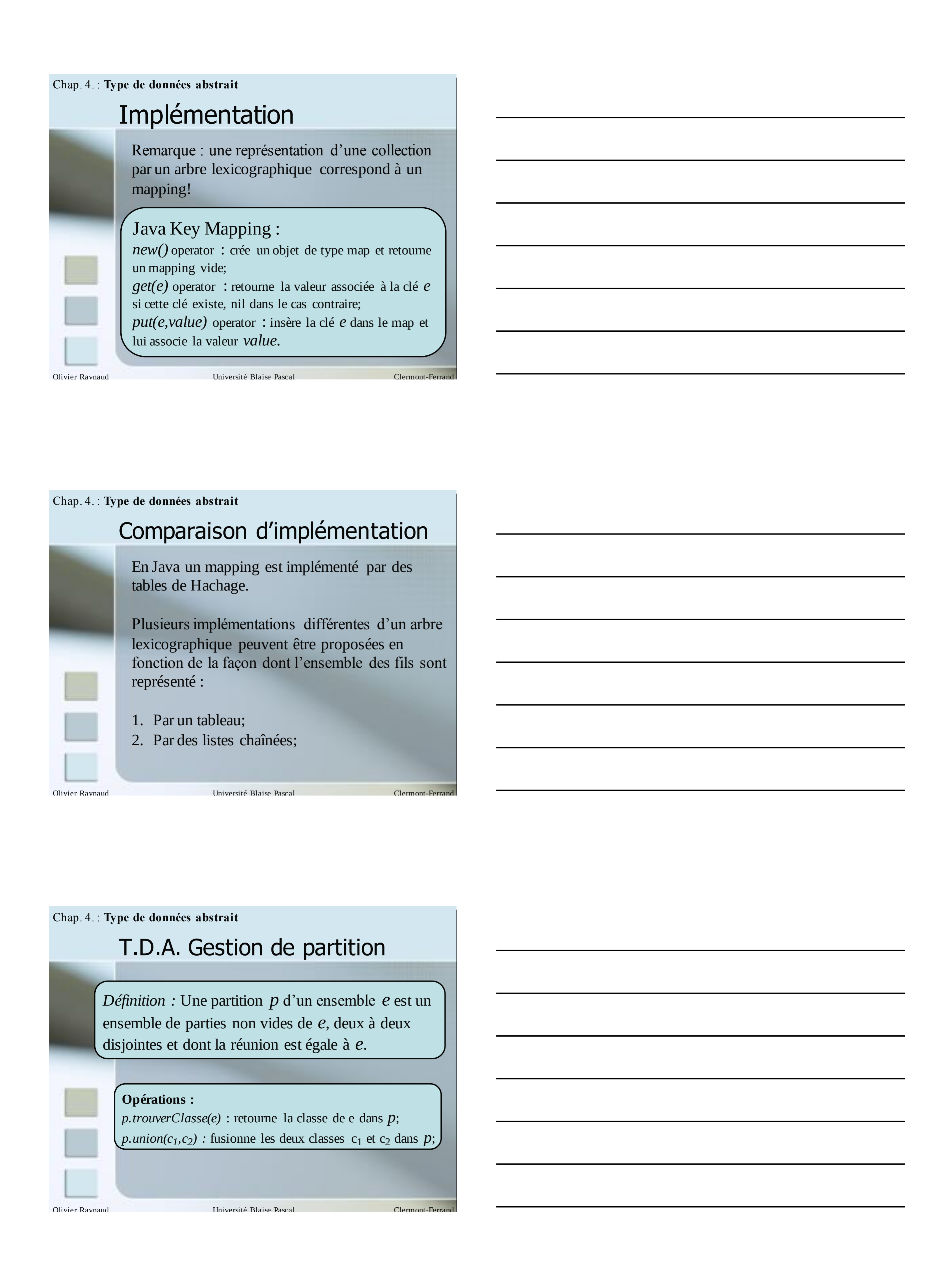
Structure de données	Insérer()	Maximum()	extraireMax()
Tableau non ordonné	O(1)	O(n)	O(n)
Liste non ordonnée	O(1)	O(n)	O(n)
Tableau ordonné	O(n)	O(1)	O(1)
Liste ordonnée	O(n)	O(1)	O(1)
Tas	à étudier	à étudier	à étudier

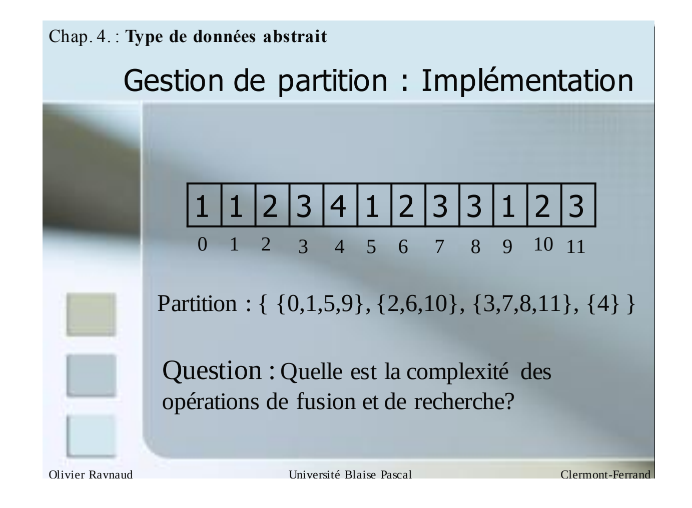


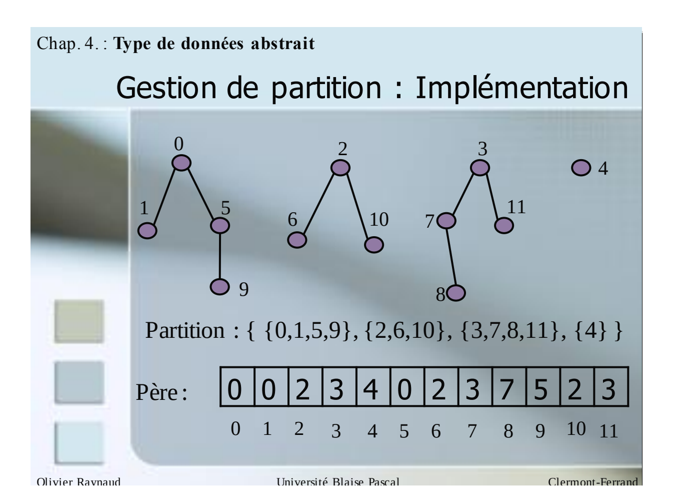

Chap. 4. : Type de données abstrait
Implémentation et complexité
Question : Quelle structure de données permettrait de proposer des algorithmes pour les opérations d'insertion, de vérification d'appartenance et de suppression admettant une complexité indépendante de la taille de la famille?
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

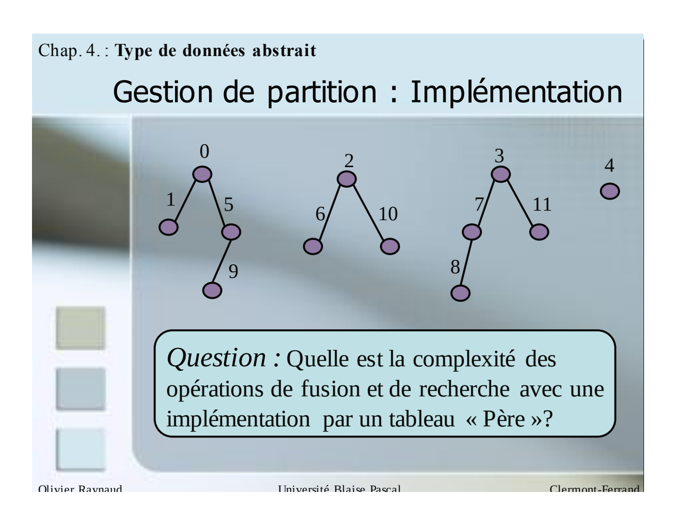

Chap. 4. : Type de données abstrait
Exemple
a
b
b
c
d
e
c
d
c
d
d
d
e
e
e
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

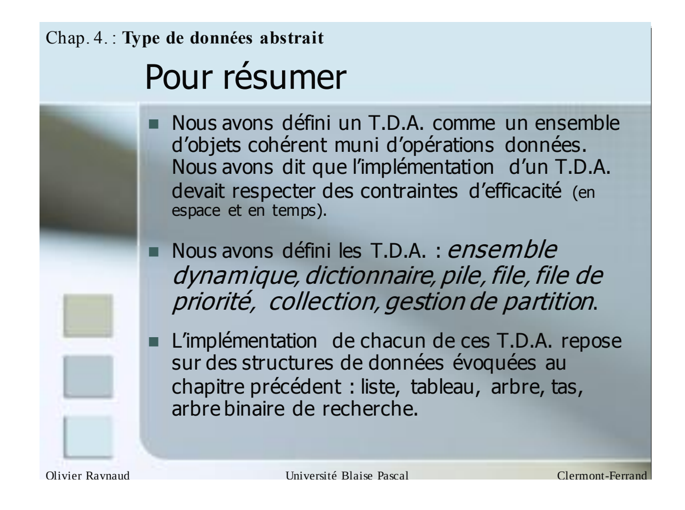

Chap. 4. : Type de données abstrait
L'arbre lexicographique
Définition : soit F une famille de sous-ensembles de X, nous associons à F un arbre T(F) lexicographique unique tel que :
1. chaque arête de l'arbre est étiqueté par un élément de X;
2. à chaque nœud notifié de l'arbre correspond un mot de F;
3. à chaque mot de F correspond un chemin unique dans l'arbre tel que ce mot corresponde à la concaténation des étiquettes de ce chemin;
4. l'ordre des arêtes d'un chemin coïncident avec l'ordre $<_x$;
5. l'ordre des arêtes sortant d'un nœud coïncident avec l'ordre $<_x$.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand


Chap. 4. : Type de données abstrait
Implémentation
Remarque : une représentation d'une collection par un arbre lexicographique correspond à un mapping!
Java Key Mapping :
new() operator : crée un objet de type map et retourne un mapping vide;
get(e) operator : retourne la valeur associée à la clé e si cette clé existe, nil dans le cas contraire;
put(e,value) operator : insère la clé e dans le map et lui associe la valeur value.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand
Chap. 4. : Type de données abstrait
Comparaison d'implémentation
En Java un mapping est implémenté par des tables de Hachage.
Plusieurs implémentations différentes d'un arbre lexicographique peuvent être proposées en fonction de la façon dont l'ensemble des fils sont représenté :
1. Par un tableau;
2. Par des listes chaînées;
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand
Chap. 4. : Type de données abstrait
T.D.A. Gestion de partition
Définition : Une partition p d'un ensemble e est un ensemble de parties non vides de e, deux à deux disjointes et dont la réunion est égale à e.
Opérations :
p.trouverClasse(e) : retourne la classe de e dans p;
p.union(c1,c2) : fusionne les deux classes c1 et c2 dans p;
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

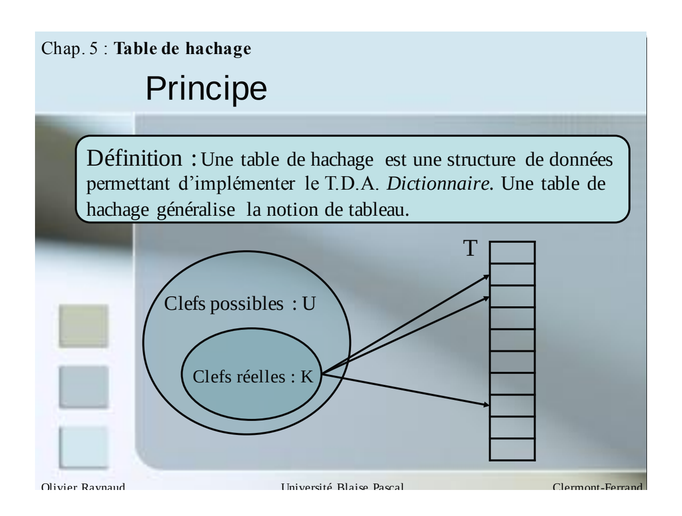

Chap. 4. : Type de données abstrait
Gestion de partition : Implémentation
1 1 2 3 4 1 2 3 3 1 2 3
0 1 2 3 4 5 6 7 8 9 10 11
Partition : { {0,1,5,9}, {2,6,10}, {3,7,8,11}, {4} }
Question : Quelle est la complexité des opérations de fusion et de recherche?
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

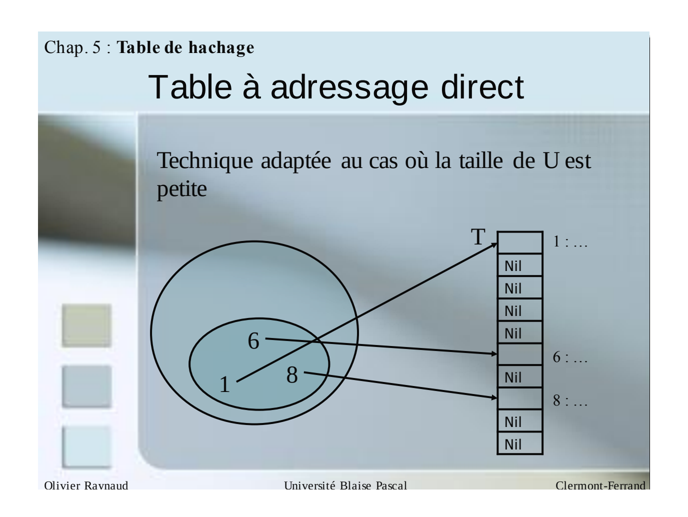

Chap. 4. : Type de données abstrait
Gestion de partition : Implémentation
0 1 5 9 2 6 10 3 7 11 8 4
Partition : { {0,1,5,9}, {2,6,10}, {3,7,8,11}, {4} }
Père : 0 0 2 3 4 0 2 3 7 5 2 3
0 1 2 3 4 5 6 7 8 9 10 11
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

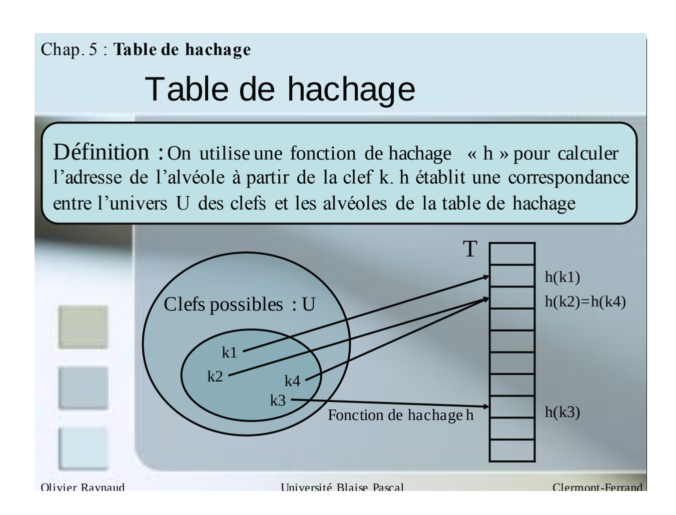

Chap. 4. : Type de données abstrait
Gestion de partition : Implémentation
0 1 5 9 2 6 10 3 7 11 8 4
Question : Quelle est la complexité des opérations de fusion et de recherche avec une implémentation par un tableau « Père »?
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

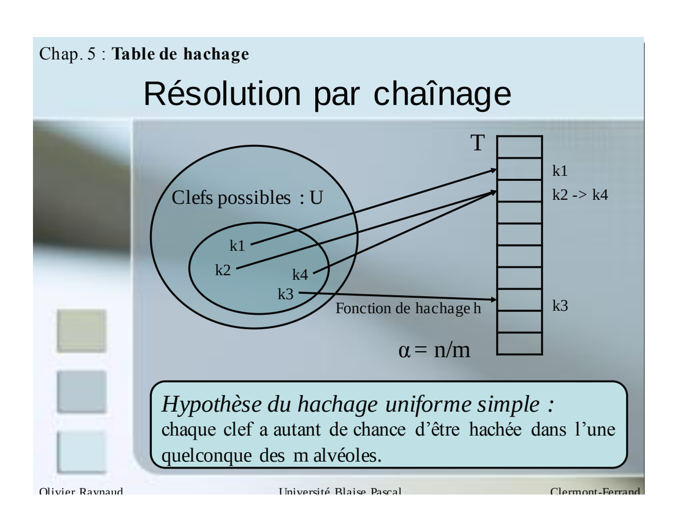

Chap. 4. : **Type de données abstrait**
Pour résumer
Nous avons défini un T.D.A. comme un ensemble d'objets cohérent muni d'opérations données. Nous avons dit que l'implémentation d'un T.D.A. devait respecter des contraintes d'efficacité (en espace et en temps).
Nous avons défini les T.D.A. : *ensemble dynamique, dictionnaire, pile, file, file de priorité, collection, gestion de partition*.
L'implémentation de chacun de ces T.D.A. repose sur des structures de données évoquées au chapitre précédent : liste, tableau, arbre, tas, arbre binaire de recherche.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

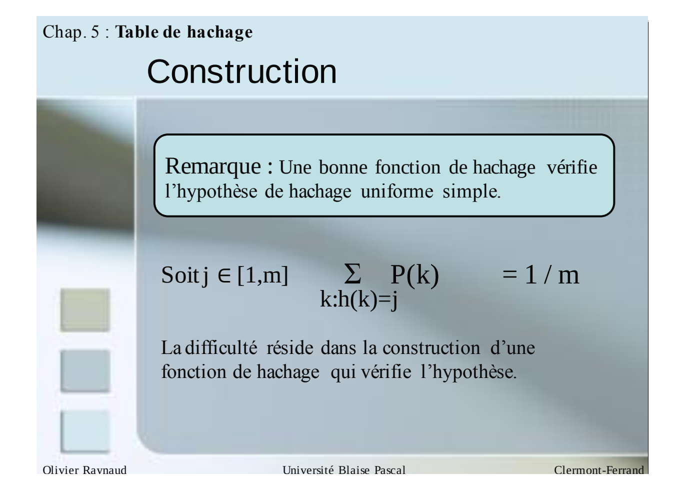

Chapitre 5
Table de hachage
Olivier Raynaud
Université Blaise Pascal

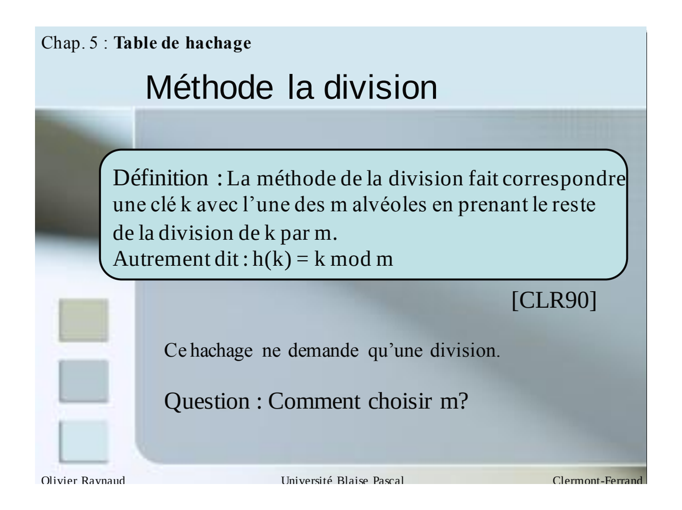

Chap. 5 : **Table de hachage**
Principe
Définition : Une table de hachage est une structure de données permettant d'implémenter le T.D.A. *Dictionnaire*. Une table de hachage généralise la notion de tableau.
T
Clefs possibles : U
Clefs réelles : K
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

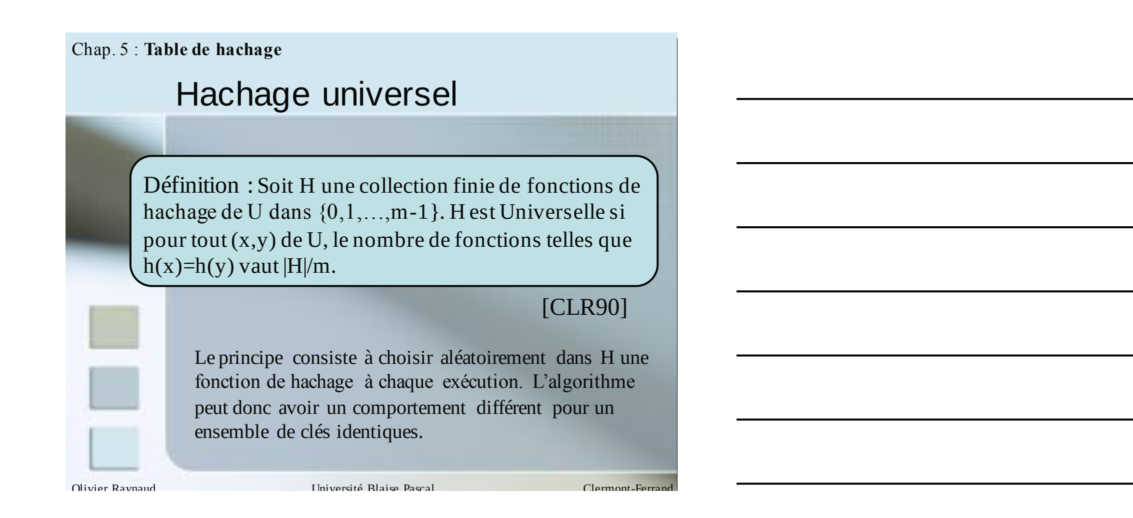

Chap. 5 : Table de hachage
Table à adressage direct
Technique adaptée au cas où la taille de U est petite
T
1 : …
Nil
Nil
Nil
Nil
6 : …
Nil
8 : …
Nil
Nil
6
1
8
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

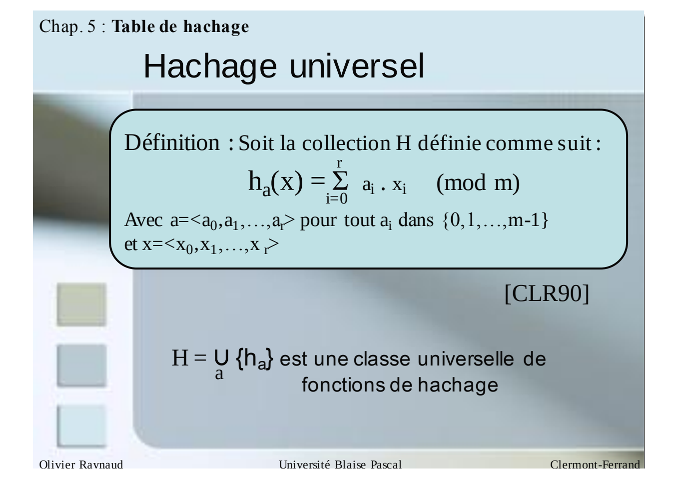

Chap. 5 : Table de hachage
Table de hachage
Définition : On utilise une fonction de hachage « h » pour calculer l'adresse de l'alvéole à partir de la clef k. h établit une correspondance entre l'univers U des clefs et les alvéoles de la table de hachage
T
Clefs possibles : U
k1
k2
k4
k3
Fonction de hachage h
h(k1)
h(k2)=h(k4)
h(k3)
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

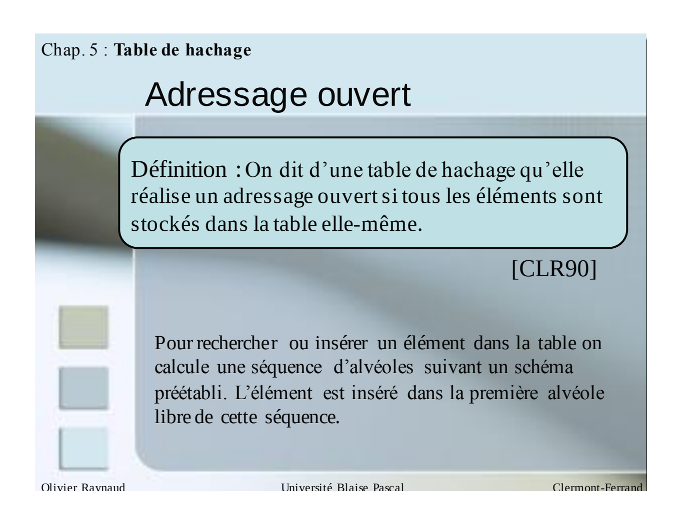

Chap. 5 : Table de hachage
Résolution par chaînage
T
Clefs possibles : U
k1
k2
k4
k3
Fonction de hachage h
k1
k2 -> k4
k3
α = n/m
Hypothèse du hachage uniforme simple :
chaque clef a autant de chance d'être hachée dans l'une quelconque des m alvéoles.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

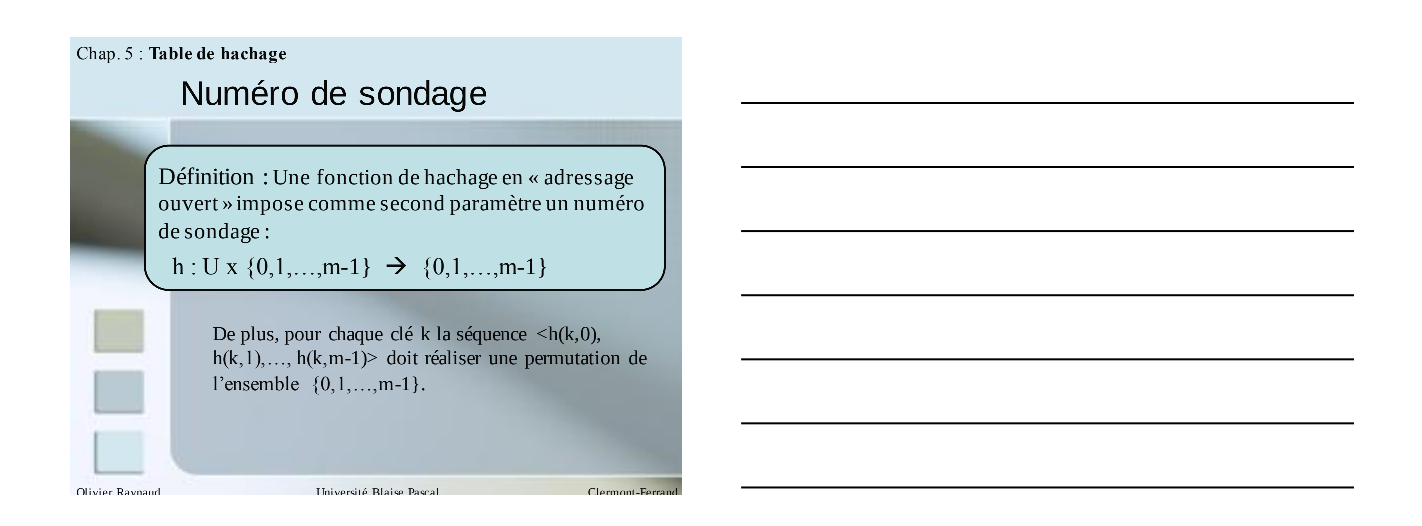

Chap. 5 : Table de hachage
Construction
Remarque : Une bonne fonction de hachage vérifie l'hypothèse de hachage uniforme simple.
Soit j ∈ [1,m] Σ P(k) = 1 / m
k:h(k)=j
La difficulté réside dans la construction d'une fonction de hachage qui vérifie l'hypothèse.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

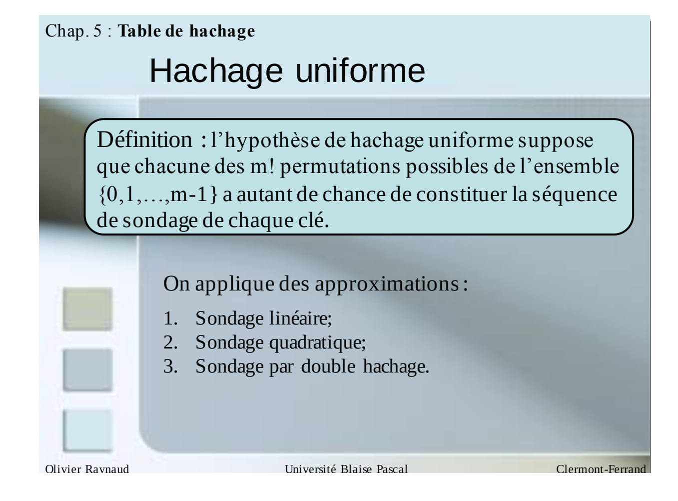

Chap. 5 : Table de hachage
Méthode la division
Définition : La méthode de la division fait correspondre une clé k avec l'une des m alvéoles en prenant le reste de la division de k par m.
Autrement dit : h(k) = k mod m
[CLR90]
Ce hachage ne demande qu'une division.
Question : Comment choisir m?
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand


Chap. 5 : Table de hachage
Hachage universel
Définition : Soit H une collection finie de fonctions de hachage de U dans {0,1,…,m-1}. H est Universelle si pour tout (x,y) de U, le nombre de fonctions telles que h(x)=h(y) vaut |H|/m.
[CLR90]
Le principe consiste à choisir aléatoirement dans H une fonction de hachage à chaque exécution. L'algorithme peut donc avoir un comportement différent pour un ensemble de clés identiques.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

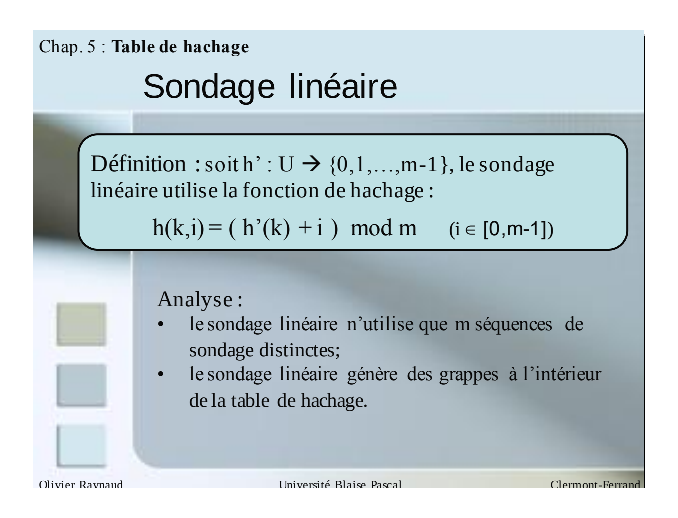

Chap. 5 : Table de hachage
Hachage universel
Définition : Soit la collection H définie comme suit :
$h_a(x) = \sum_{i=0}^{r} a_i . x_i \quad (\text{mod } m)$
Avec $a=<a_0,a_1,\ldots,a_r>$ pour tout a_i dans $\{0,1,\ldots,m-1\}$
et $x=<x_0,x_1,\ldots,x_r>$
[CLR90]
$H = \bigcup_a \{h_a\}$ est une classe universelle de fonctions de hachage
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand


Chap. 5 : Table de hachage
Adressage ouvert
Définition : On dit d'une table de hachage qu'elle réalise un adressage ouvert si tous les éléments sont stockés dans la table elle-même.
[CLR90]
Pour rechercher ou insérer un élément dans la table on calcule une séquence d'alvéoles suivant un schéma préétabli. L'élément est inséré dans la première alvéole libre de cette séquence.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

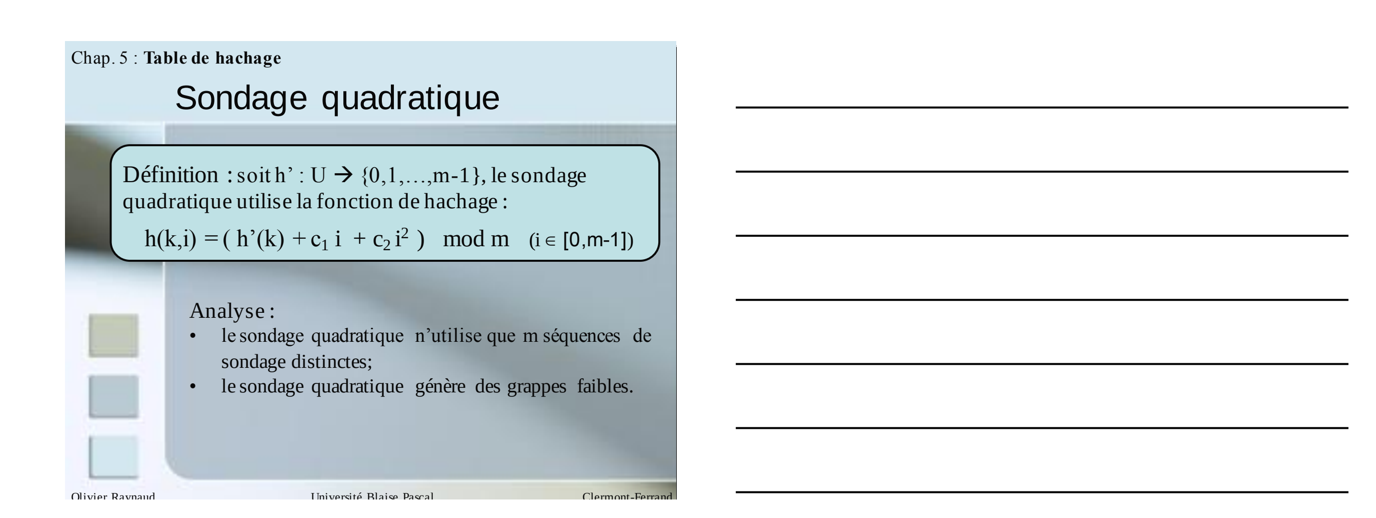

Chap. 5 : Table de hachage
Numéro de sondage
Définition : Une fonction de hachage en « adressage ouvert » impose comme second paramètre un numéro de sondage :
$h : U \times \{0,1,\ldots,m-1\} \rightarrow \{0,1,\ldots,m-1\}$
De plus, pour chaque clé k la séquence $<h(k,0), h(k,1),\ldots, h(k,m-1)>$ doit réaliser une permutation de l'ensemble $\{0,1,\ldots,m-1\}$.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand



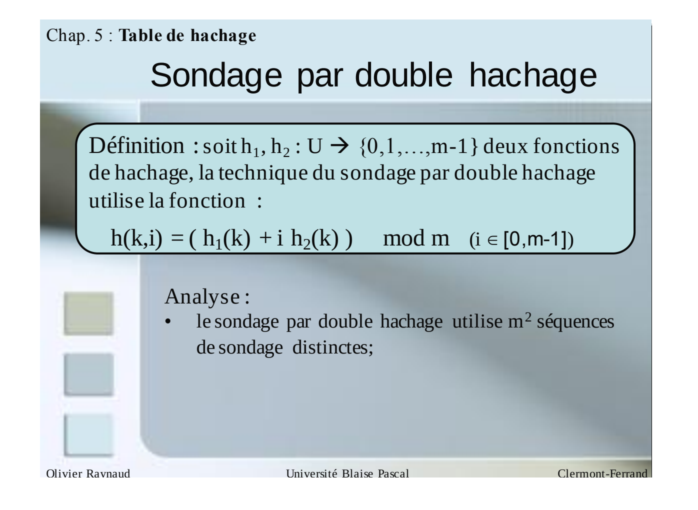
Chap. 5 : Table de hachage
Hachage uniforme
Définition : l'hypothèse de hachage uniforme suppose que chacune des m! permutations possibles de l'ensemble {0,1,…,m-1} a autant de chance de constituer la séquence de sondage de chaque clé.
On applique des approximations :
1. Sondage linéaire;
2. Sondage quadratique;
3. Sondage par double hachage.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

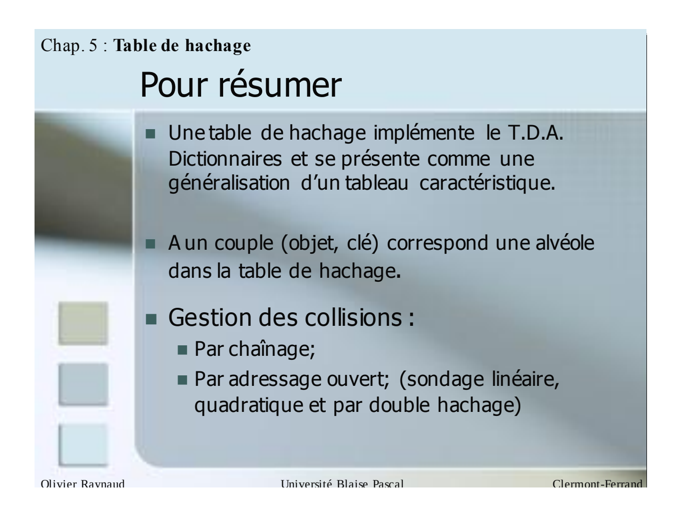


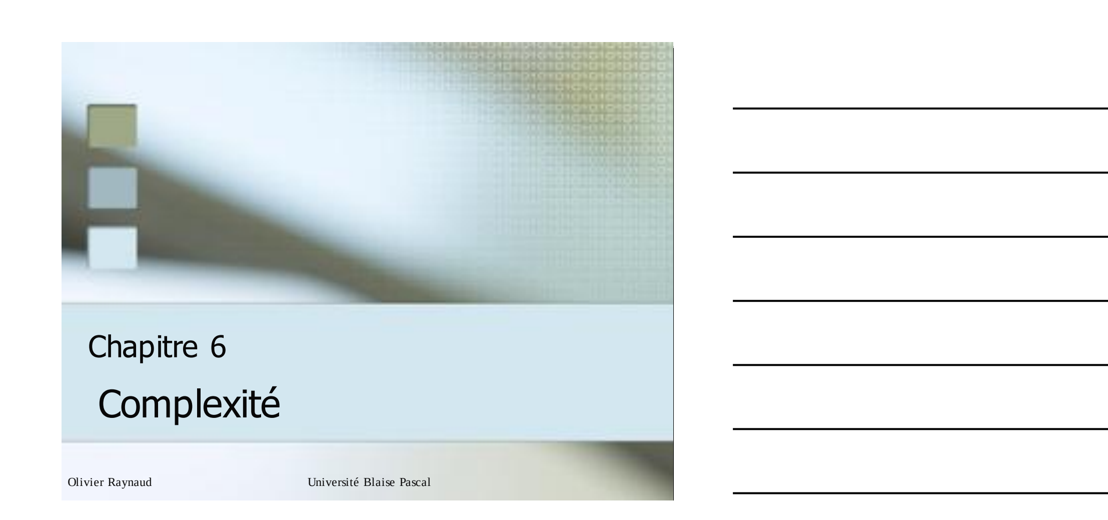
Chap. 5 : Table de hachage
Sondage linéaire
Définition : soit h' : U → {0,1,…,m-1}, le sondage linéaire utilise la fonction de hachage :
h(k,i) = (h'(k) + i) mod m (i ∈ [0,m-1])
Analyse :
• le sondage linéaire n'utilise que m séquences de sondage distinctes;
• le sondage linéaire génère des grappes à l'intérieur de la table de hachage.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

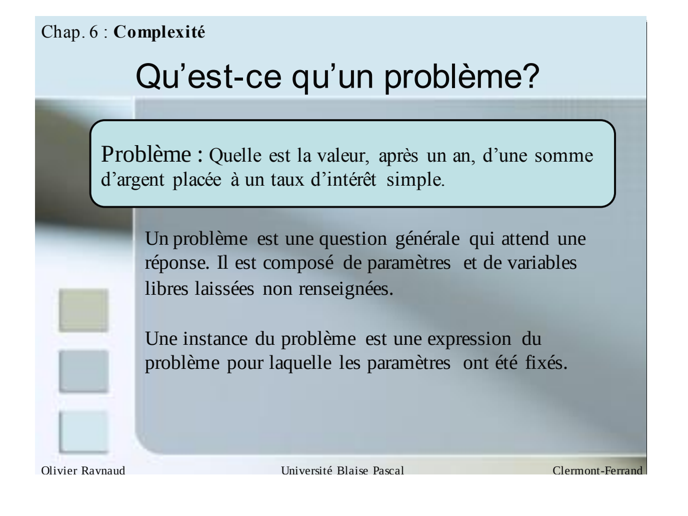


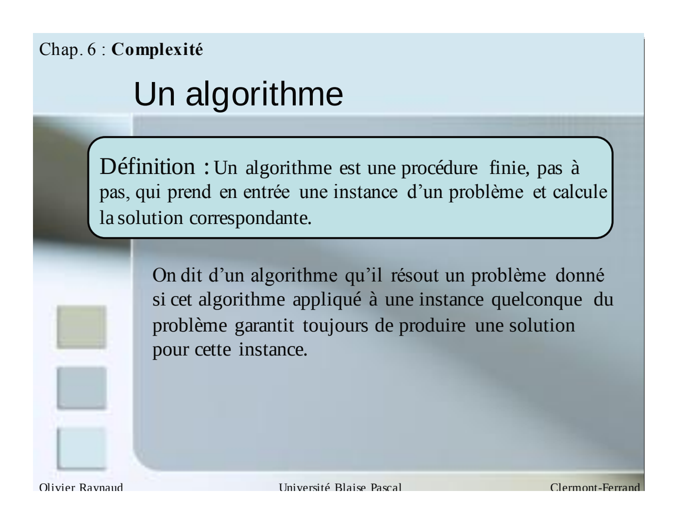
Chap. 5 : Table de hachage
Sondage quadratique
Définition : soit h' : U → {0,1,…,m-1}, le sondage quadratique utilise la fonction de hachage :
h(k,i) = (h'(k) + c1 i + c2 i2) mod m (i ∈ [0,m-1])
Analyse :
• le sondage quadratique n'utilise que m séquences de sondage distinctes;
• le sondage quadratique génère des grappes faibles.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

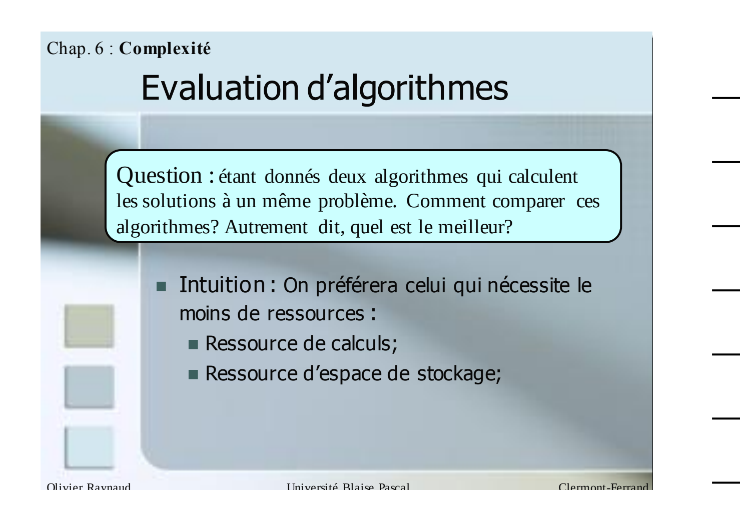

Chap. 5 : Table de hachage
Sondage par double hachage
Définition : soit h_1, h_2 : U → {0,1,…,m-1} deux fonctions de hachage, la technique du sondage par double hachage utilise la fonction :
$h(k,i) = (h_1(k) + i\, h_2(k)) \mod m$ ($i \in [0,m-1]$)
Analyse :
• le sondage par double hachage utilise m^2 séquences de sondage distinctes;
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

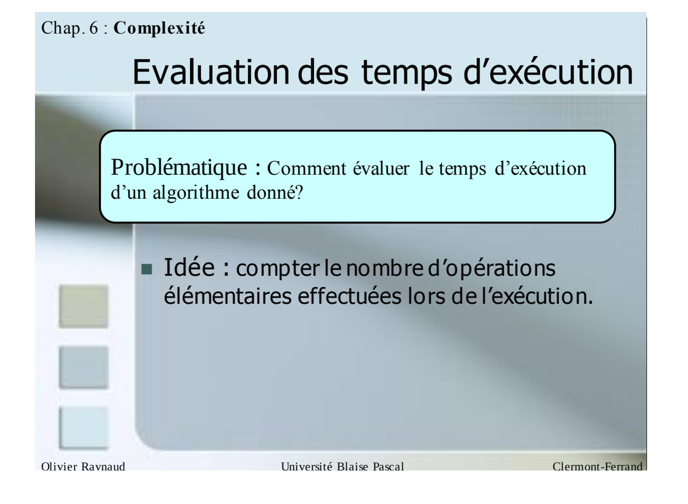

Chap. 5 : Table de hachage
Pour résumer
Une table de hachage implémente le T.D.A. Dictionnaires et se présente comme une généralisation d'un tableau caractéristique.
A un couple (objet, clé) correspond une alvéole dans la table de hachage.
Gestion des collisions :
Par chaînage;
Par adressage ouvert; (sondage linéaire, quadratique et par double hachage)
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

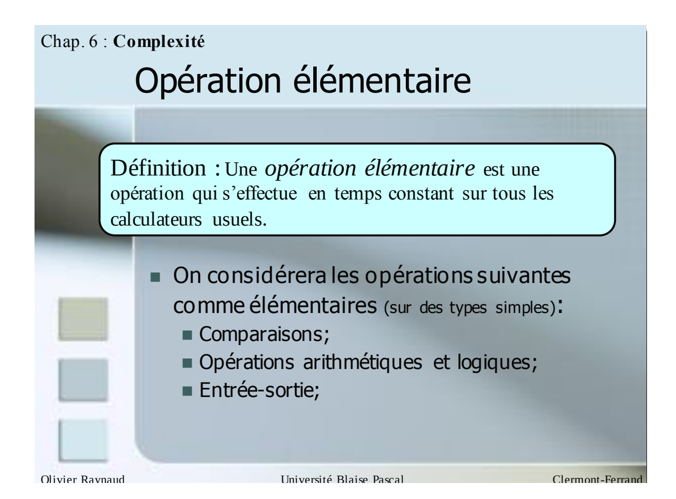

Chapitre 6
Complexité
Olivier Raynaud
Université Blaise Pascal

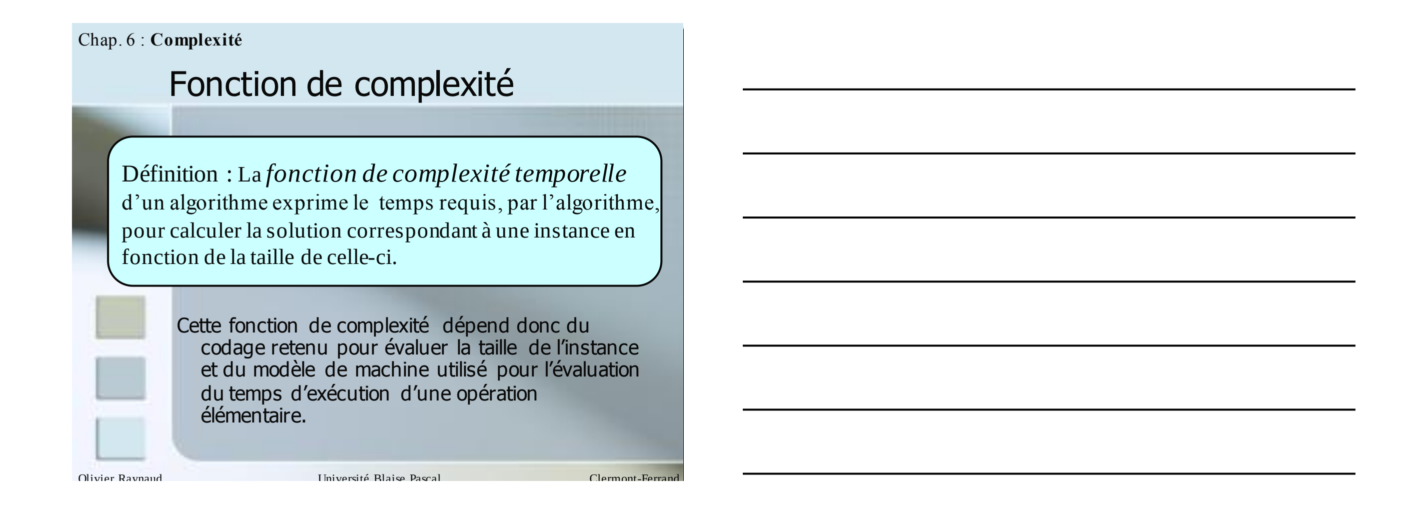

Chap. 6 : Complexité
Qu'est-ce qu'un problème?
Problème : Quelle est la valeur, après un an, d'une somme d'argent placée à un taux d'intérêt simple.
Un problème est une question générale qui attend une réponse. Il est composé de paramètres et de variables libres laissées non renseignées.
Une instance du problème est une expression du problème pour laquelle les paramètres ont été fixés.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

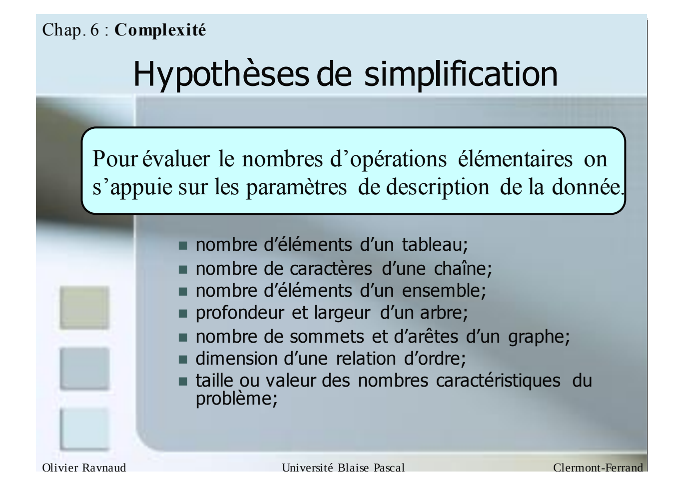

Chap. 6 : Complexité
Un algorithme
Définition : Un algorithme est une procédure finie, pas à pas, qui prend en entrée une instance d'un problème et calcule la solution correspondante.
On dit d'un algorithme qu'il résout un problème donné si cet algorithme appliqué à une instance quelconque du problème garantit toujours de produire une solution pour cette instance.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

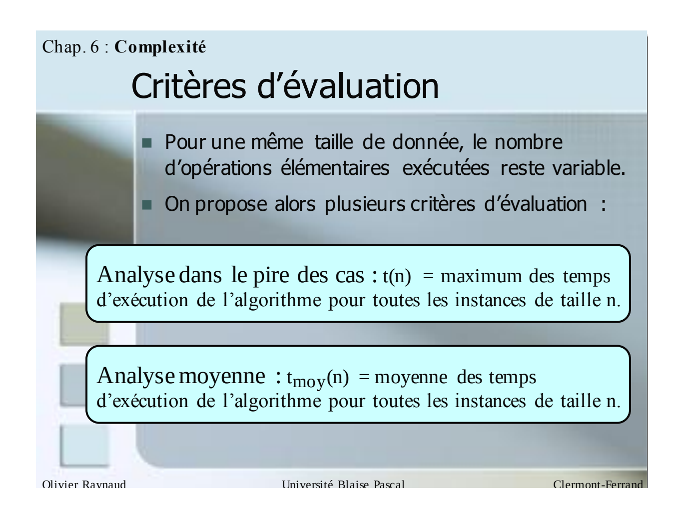

Chap. 6 : Complexité
Evaluation d'algorithmes
Question : étant donnés deux algorithmes qui calculent les solutions à un même problème. Comment comparer ces algorithmes? Autrement dit, quel est le meilleur?
Intuition : On préférera celui qui nécessite le moins de ressources :
Ressource de calculs;
Ressource d'espace de stockage;
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

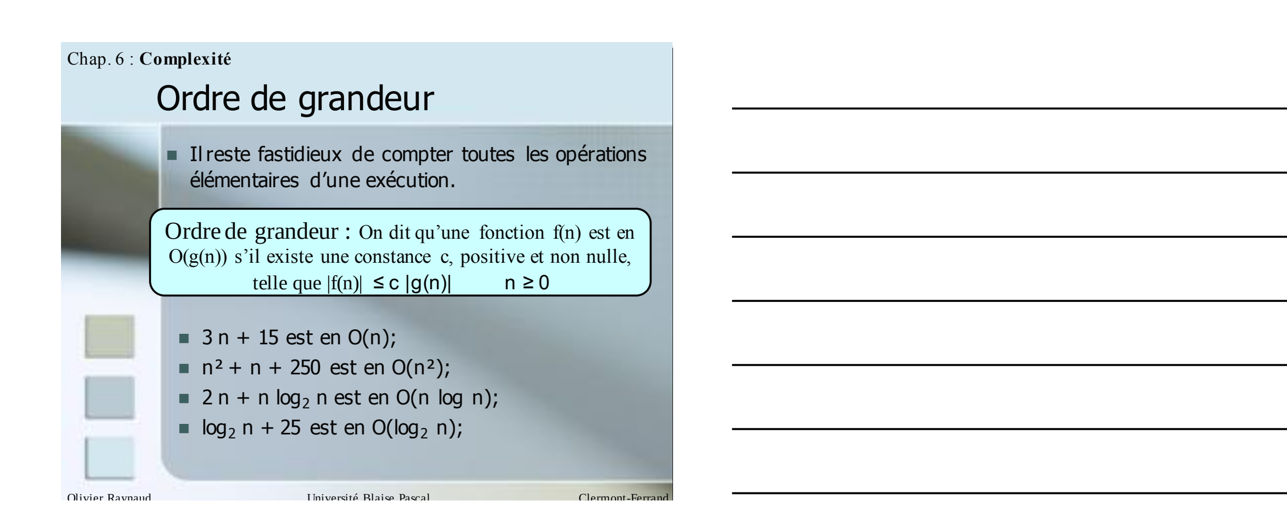

Chap. 6 : Complexité
Evaluation des temps d'exécution
Problématique : Comment évaluer le temps d'exécution d'un algorithme donné?
Idée : compter le nombre d'opérations élémentaires effectuées lors de l'exécution.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

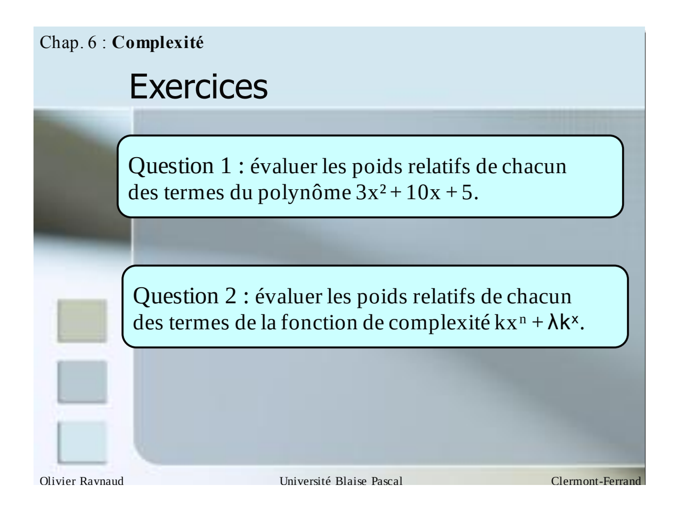

Chap. 6 : Complexité
Opération élémentaire
Définition : Une opération élémentaire est une opération qui s'effectue en temps constant sur tous les calculateurs usuels.
On considérera les opérations suivantes comme élémentaires (sur des types simples):
Comparaisons;
Opérations arithmétiques et logiques;
Entrée-sortie;
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

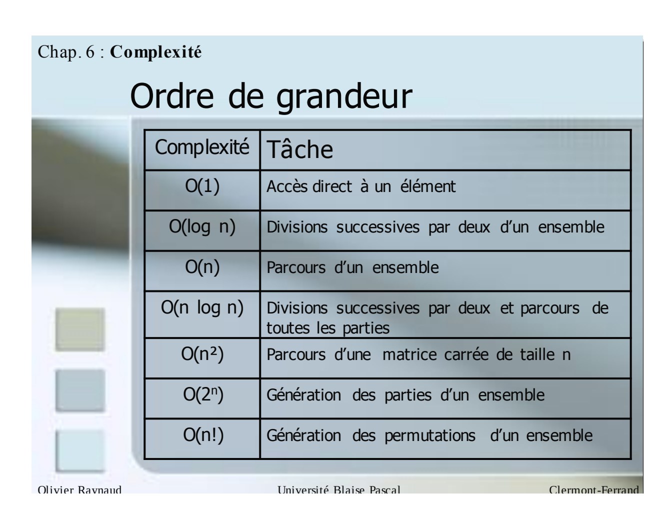

Chap. 6 : Complexité
Fonction de complexité
Définition : La fonction de complexité temporelle d'un algorithme exprime le temps requis, par l'algorithme, pour calculer la solution correspondant à une instance en fonction de la taille de celle-ci.
Cette fonction de complexité dépend donc du codage retenu pour évaluer la taille de l'instance et du modèle de machine utilisé pour l'évaluation du temps d'exécution d'une opération élémentaire.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand


Chap. 6 : Complexité
Hypothèses de simplification
Pour évaluer le nombres d'opérations élémentaires on s'appuie sur les paramètres de description de la donnée.
nombre d'éléments d'un tableau;
nombre de caractères d'une chaîne;
nombre d'éléments d'un ensemble;
profondeur et largeur d'un arbre;
nombre de sommets et d'arêtes d'un graphe;
dimension d'une relation d'ordre;
taille ou valeur des nombres caractéristiques du problème;
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

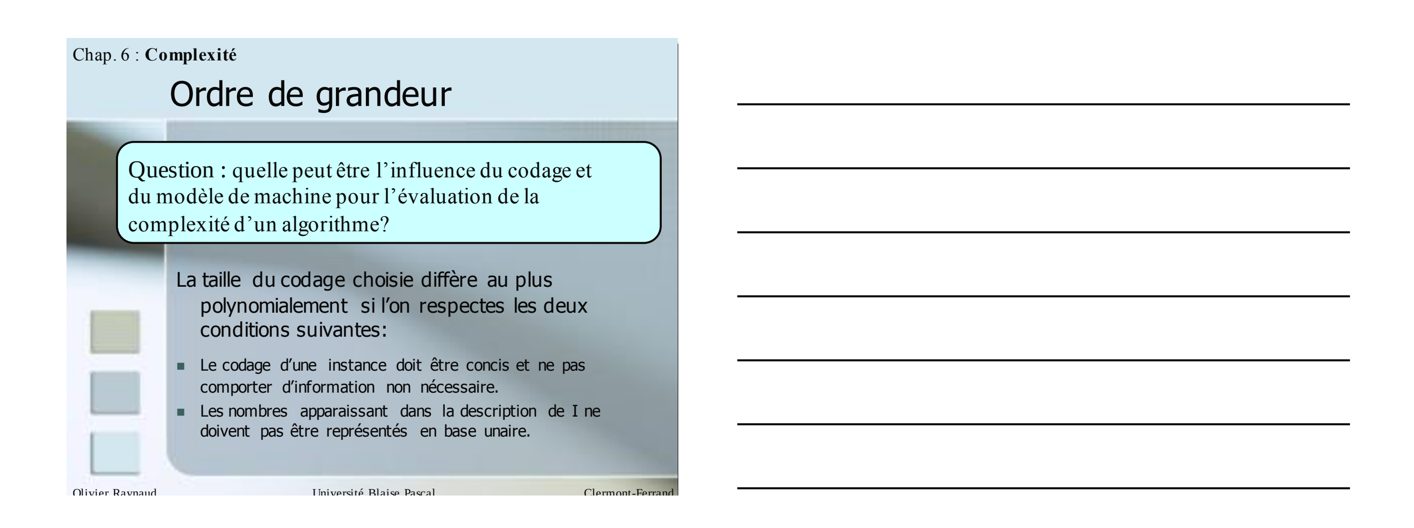

Chap. 6 : Complexité
Critères d'évaluation
Pour une même taille de donnée, le nombre d'opérations élémentaires exécutées reste variable.
On propose alors plusieurs critères d'évaluation :
Analyse dans le pire des cas : t(n) = maximum des temps d'exécution de l'algorithme pour toutes les instances de taille n.
Analyse moyenne : $t_{moy}(n)$ = moyenne des temps d'exécution de l'algorithme pour toutes les instances de taille n.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

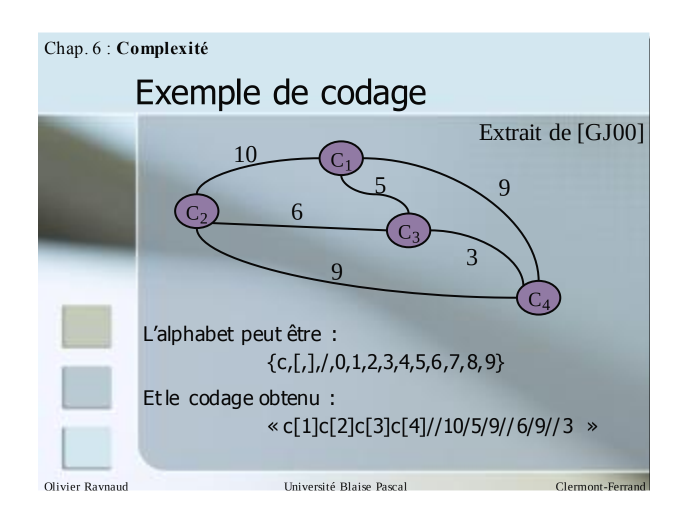

Chap. 6 : Complexité
Ordre de grandeur
Il reste fastidieux de compter toutes les opérations élémentaires d'une exécution.
Ordre de grandeur : On dit qu'une fonction f(n) est en O(g(n)) s'il existe une constance c, positive et non nulle, telle que $|f(n)| \leq c\,|g(n)|$ $n \geq 0$
$3n + 15$ est en $O(n)$;
$n^2 + n + 250$ est en $O(n^2)$;
$2n + n \log_2 n$ est en $O(n \log n)$;
$\log_2 n + 25$ est en $O(\log_2 n)$;
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

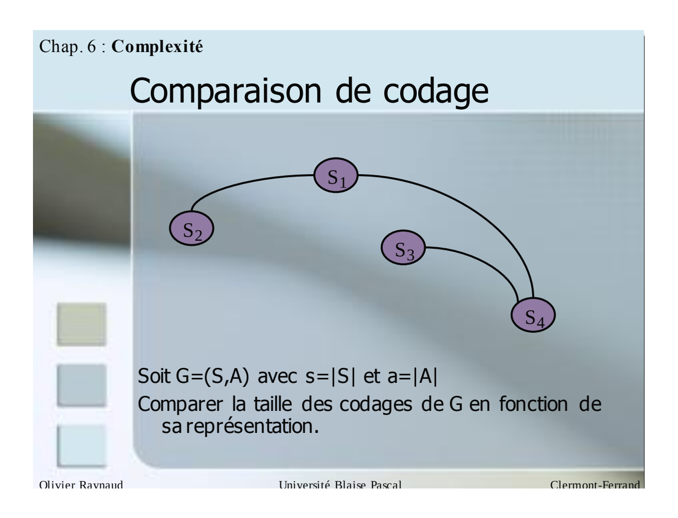




Complexité	Tâche
O(1)	Accès direct à un élément
O(log n)	Divisions successives par deux d'un ensemble
O(n)	Parcours d'un ensemble
O(n log n)	Divisions successives par deux et parcours de toutes les parties
O(n²)	Parcours d'une matrice carrée de taille n
O(2ⁿ)	Génération des parties d'un ensemble
O(n!)	Génération des permutations d'un ensemble







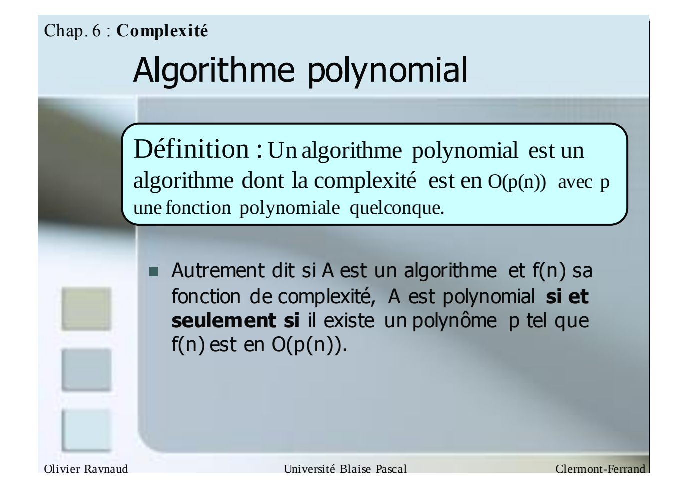
Impact du modèle de machine

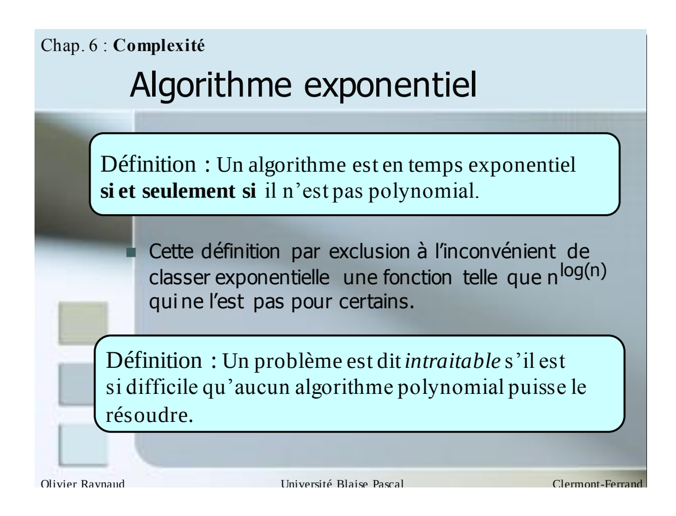
Machine simulant

Machine simulée

	1TM	**kTM**	**RAM**
1TM		$O(T(n))$	$O(T(n)\log T(n))$
kTM	$O(T^2(n))$		$O(T(n)\log T(n))$
RAM	$O(T^3(n))$	$O(T^2(n))$	

Extrait de [GJ00]





Chap. 6 : **Complexité**

Extrait de [GJ00]

Temps d'exécution

fonction	10	20	30	40	50	60
n	.00001s	.00002s	.00003s	.00004s	.00005s	.00006s
n^2	.0001s	.0004s	.0009s	.0016s	.0025	.0036s
n^3	.001s	.008s	.027s	.064s	.125s	.216s
n^5	.1s	3.2s	24.3s	1.7mn	5.2mn	13.0mn
2^n	.001s	1.0s	17.9mn	12.7j	35.7an	366siè
3^n	.059s	58 mn	6.5 an	3855siè	2 10^8siè	1.3$10^{13}$siè

Olivier Raynaud Université Blaise Pascal Clermont-Ferrand

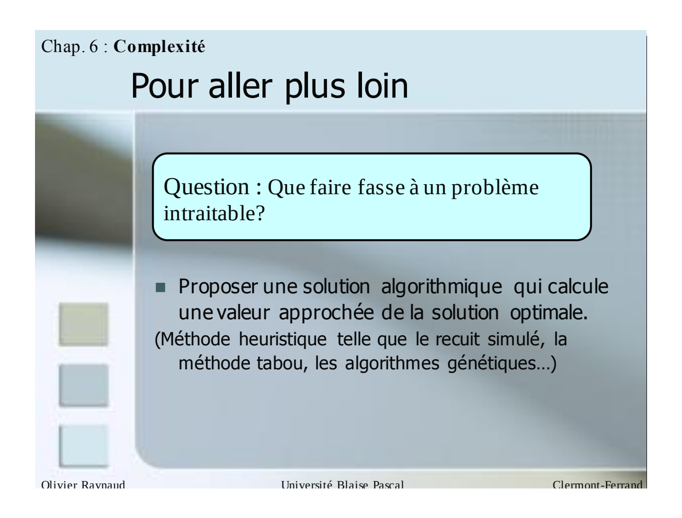
Chap. 6 : **Complexité**

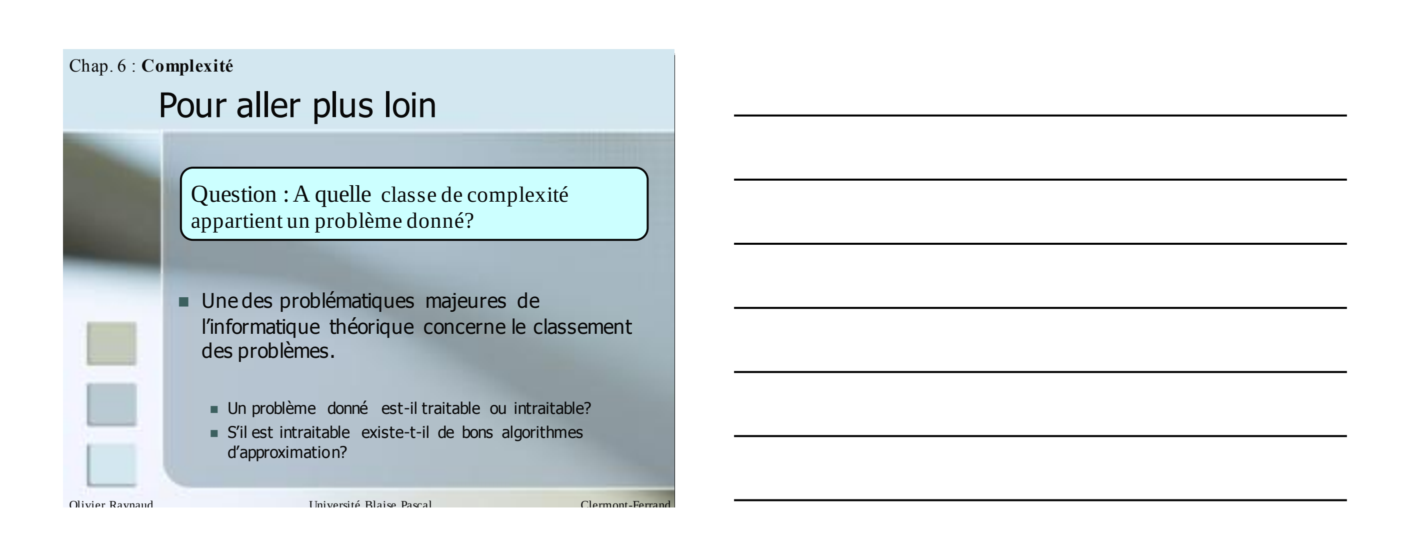
Extrait de [GJ00]

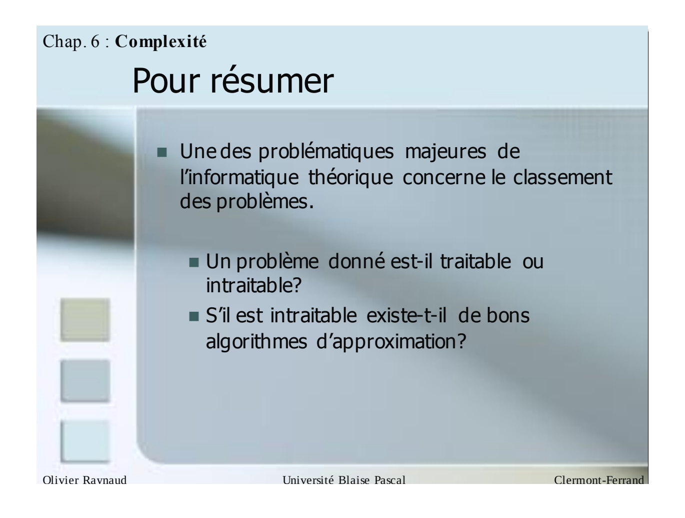
Projection

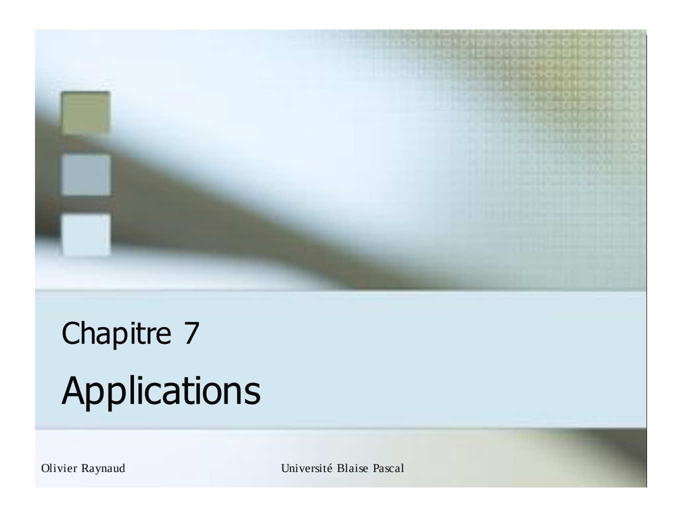
fonction	1	100	1000
n	N_1	$100N_1$	$1000N_1$
n^2	N_2	$10N_2$	$31.6N_2$
n^3	N_3	$4.63N_3$	$10N_3$
n^5	N_4	$2.5N_4$	$3.98N_4$
2^n	N_5	$N_5+6.64$	$N_5+9.97$
3^n	N_6	$N_6+4.19$	$N_6+6.29$

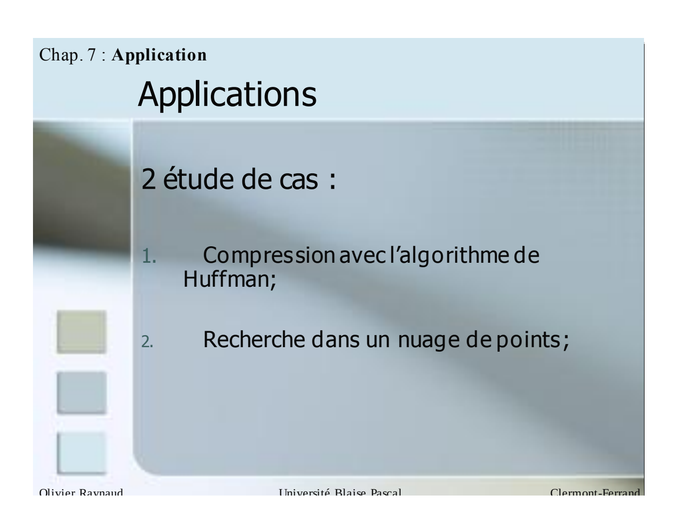
Olivier Raynaud Université Blaise Pascal Clermont-Ferrand

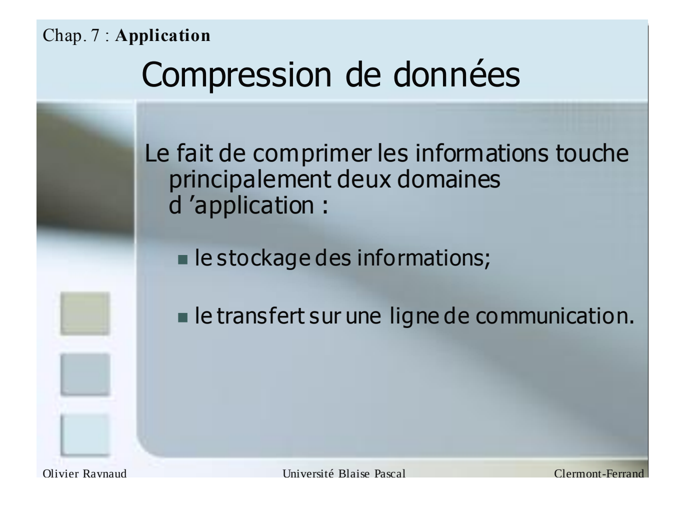


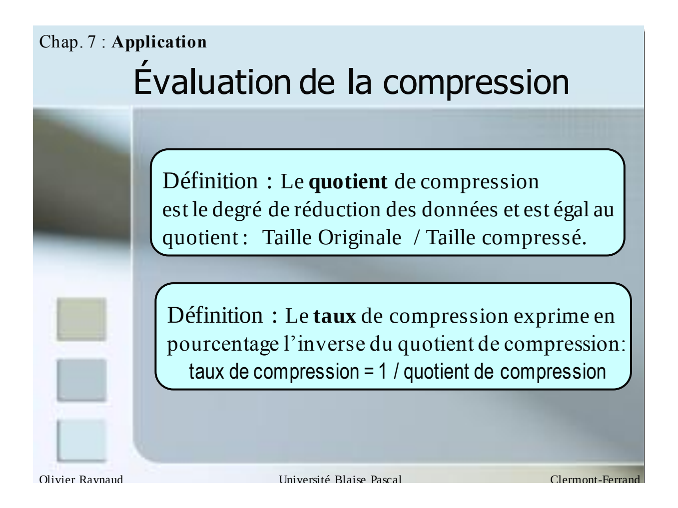


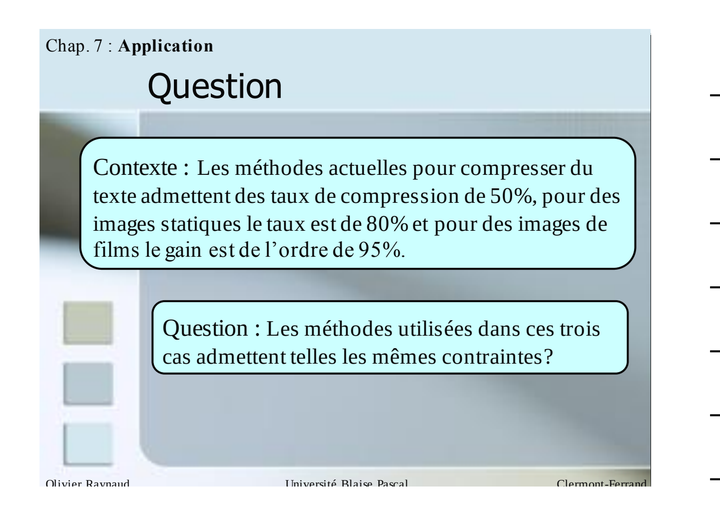

Chap. 6 : Complexité
Pour résumer
Une des problématiques majeures de l'informatique théorique concerne le classement des problèmes.
Un problème donné est-il traitable ou intraitable?
S'il est intraitable existe-t-il de bons algorithmes d'approximation?
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

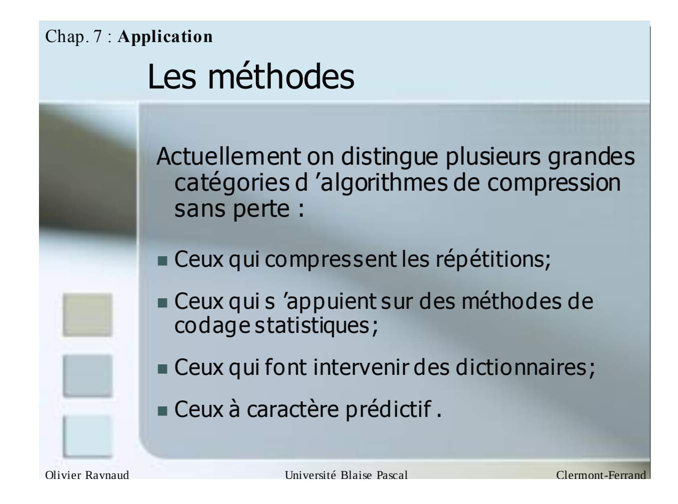

Chapitre 7
Applications
Olivier Raynaud
Université Blaise Pascal

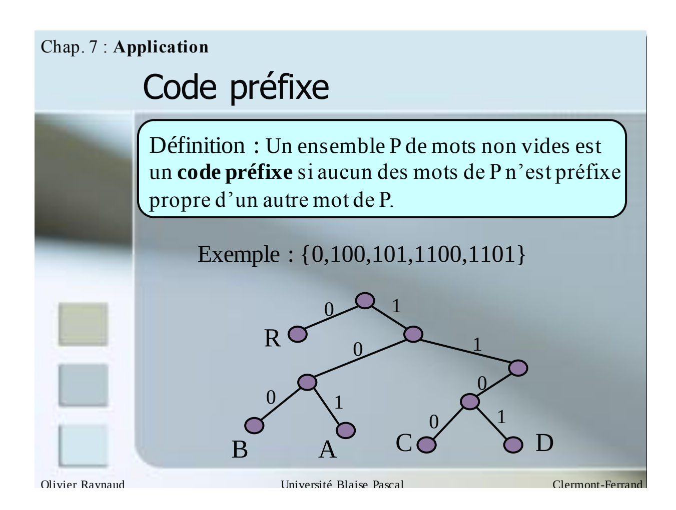

Chap. 7 : Application
Applications
2 étude de cas :
1. Compression avec l'algorithme de Huffman;
2. Recherche dans un nuage de points;
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

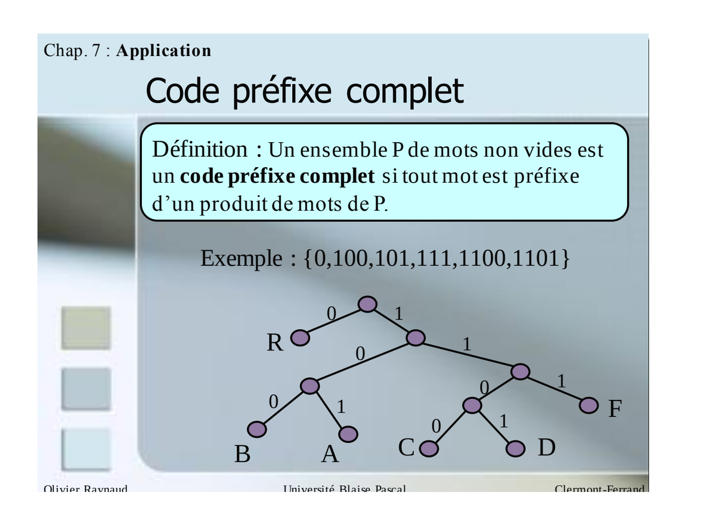

Chap. 7 : Application
Compression de données
Le fait de comprimer les informations touche principalement deux domaines d 'application :
le stockage des informations;
le transfert sur une ligne de communication.
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

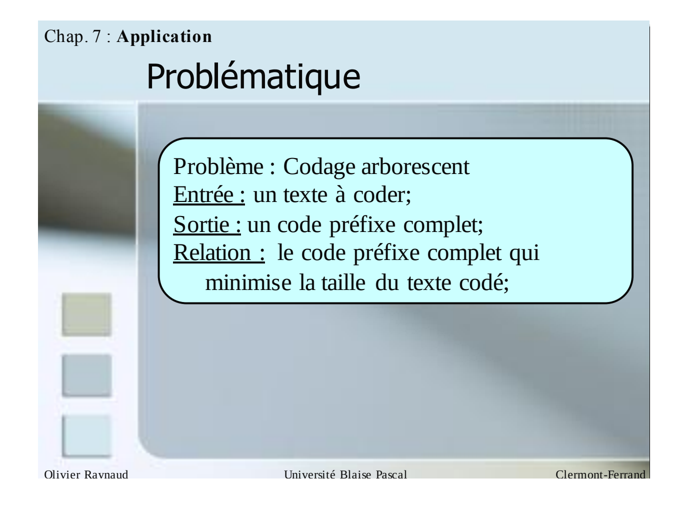

Chap. 7 : Application
Évaluation de la compression
Définition : Le quotient de compression est le degré de réduction des données et est égal au quotient : Taille Originale / Taille compressé.
Définition : Le taux de compression exprime en pourcentage l'inverse du quotient de compression:
taux de compression = 1 / quotient de compression
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

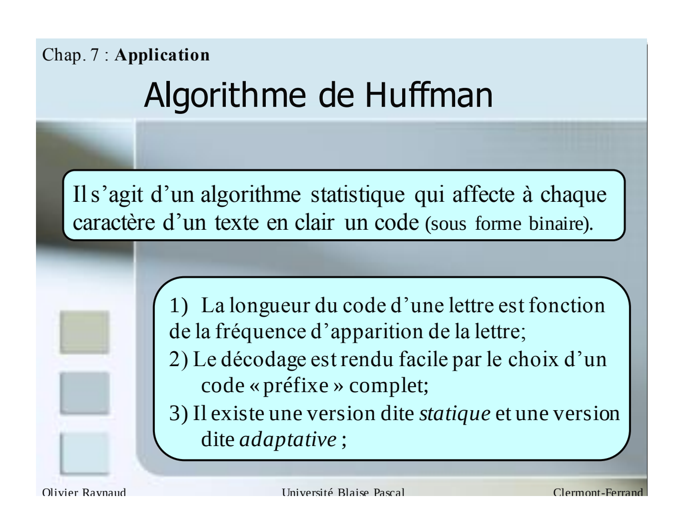

Chap. 7 : Application
Question
Contexte : Les méthodes actuelles pour compresser du texte admettent des taux de compression de 50%, pour des images statiques le taux est de 80% et pour des images de films le gain est de l'ordre de 95%.
Question : Les méthodes utilisées dans ces trois cas admettent telles les mêmes contraintes?
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

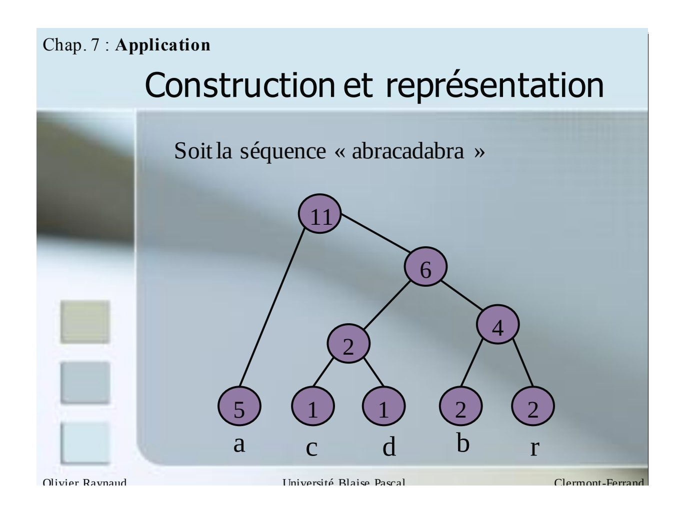

Chap. 7 : Application
Les méthodes
Actuellement on distingue plusieurs grandes catégories d 'algorithmes de compression sans perte :
Ceux qui compressent les répétitions;
Ceux qui s 'appuient sur des méthodes de codage statistiques;
Ceux qui font intervenir des dictionnaires;
Ceux à caractère prédictif .
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

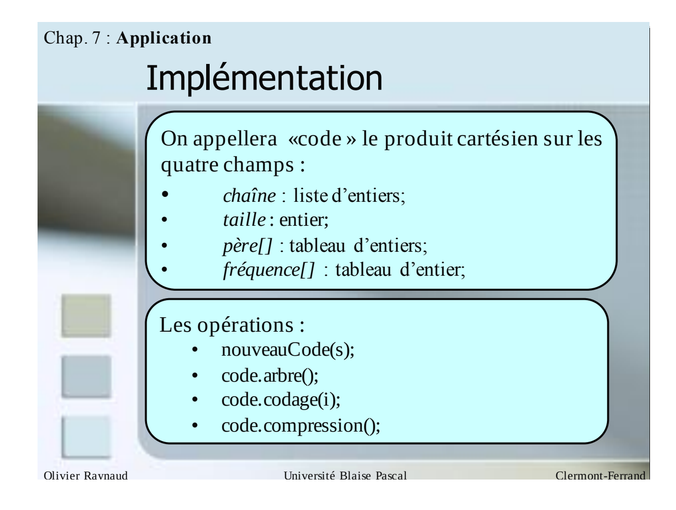

Chap. 7 : Application
Code préfixe
Définition : Un ensemble P de mots non vides est un code préfixe si aucun des mots de P n'est préfixe propre d'un autre mot de P.
Exemple : {0,100,101,1100,1101}
R
B
A
C
D
0
1
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand


Chap. 7 : Application
Code préfixe complet
Définition : Un ensemble P de mots non vides est un code préfixe complet si tout mot est préfixe d'un produit de mots de P.
Exemple : {0,100,101,111,1100,1101}
R
B
A
C
D
F
0
1
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

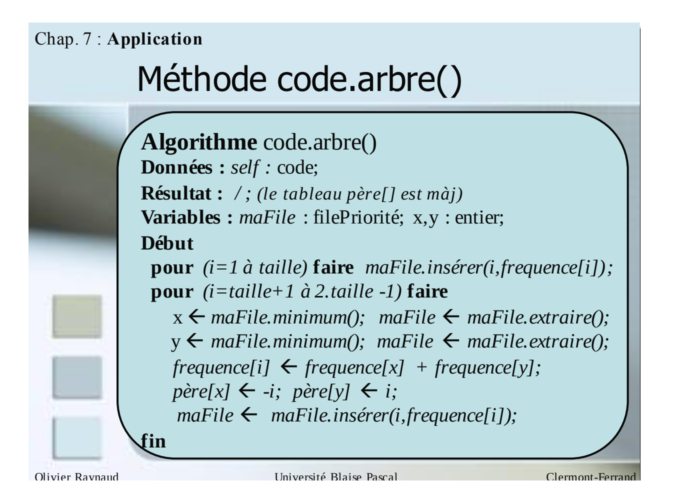

Chap. 7 : Application
Problématique
Problème : Codage arborescent
Entrée : un texte à coder;
Sortie : un code préfixe complet;
Relation : le code préfixe complet qui minimise la taille du texte codé;
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand


Chap. 7 : Application
Algorithme de Huffman
Il s'agit d'un algorithme statistique qui affecte à chaque caractère d'un texte en clair un code (sous forme binaire).
1) La longueur du code d'une lettre est fonction de la fréquence d'apparition de la lettre;
2) Le décodage est rendu facile par le choix d'un code « préfixe » complet;
3) Il existe une version dite statique et une version dite adaptative ;
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

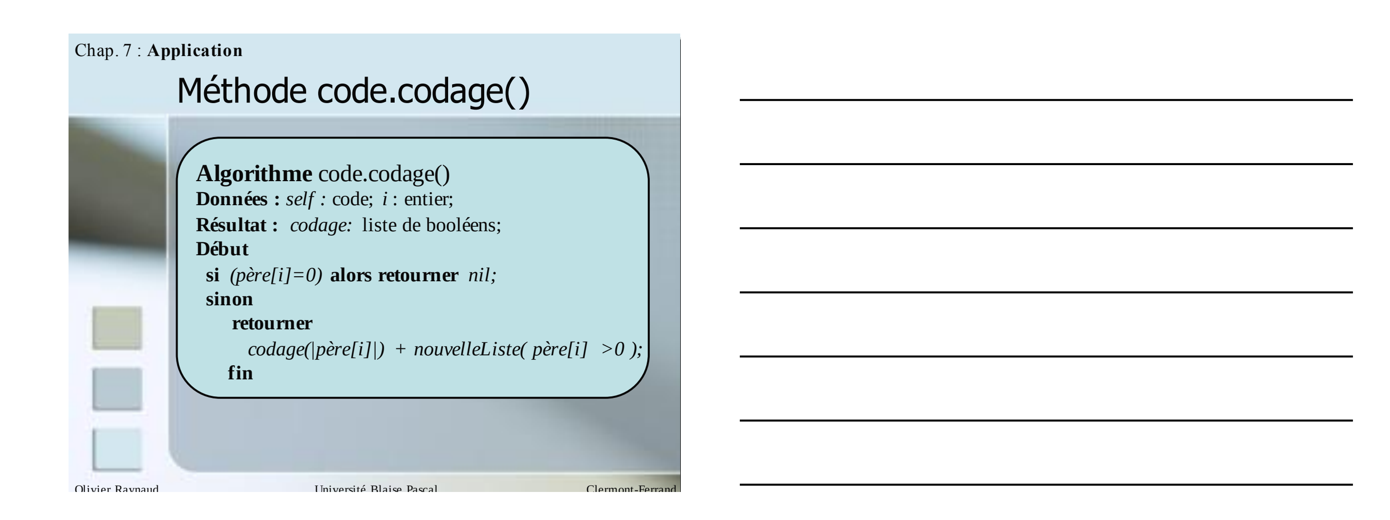

Chap. 7 : Application
Construction et représentation
Soit la séquence « abracadabra »
11
6
2
4
5
1
1
2
2
a
c
d
b
r
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand



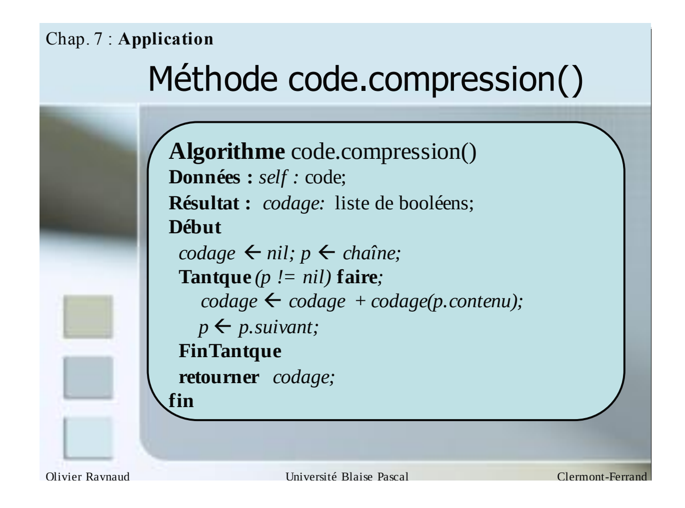
Chap. 7 : Application
Implémentation
On appellera «code » le produit cartésien sur les quatre champs :
• chaîne : liste d'entiers;
• taille : entier;
• père[] : tableau d'entiers;
• fréquence[] : tableau d'entier;
Les opérations :
• nouveauCode(s);
• code.arbre();
• code.codage(i);
• code.compression();
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand

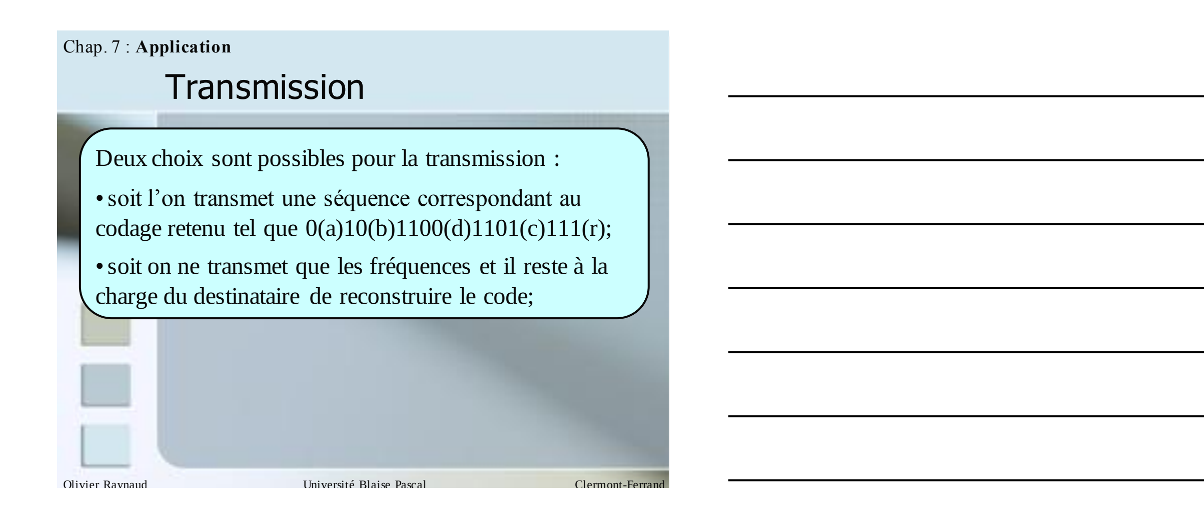


Chap. 7 : Application
Méthode code.arbre()
Algorithme code.arbre()
Données : self : code;
Résultat : / ; (le tableau père[] est màj)
Variables : maFile : filePriorité; x,y : entier;
Début
pour (i=1 à taille) faire maFile.insérer(i,frequence[i]);
pour (i=taille+1 à 2.taille -1) faire
x ← maFile.minimum(); maFile ← maFile.extraire();
y ← maFile.minimum(); maFile ← maFile.extraire();
frequence[i] ← frequence[x] + frequence[y];
père[x] ← -i; père[y] ← i;
maFile ← maFile.insérer(i,frequence[i]);
fin
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand



Chap. 7 : Application
Méthode code.codage()
Algorithme code.codage()
Données : self : code; i : entier;
Résultat : codage: liste de booléens;
Début
si (père[i]=0) alors retourner nil;
sinon
retourner
codage(|père[i]|) + nouvelleListe(père[i] >0);
fin
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand


Chap. 7 : Application
Méthode code.compression()
Algorithme code.compression()
Données : self : code;
Résultat : codage: liste de booléens;
Début
codage ← nil; p ← chaîne;
Tantque (p != nil) faire;
codage ← codage + codage(p.contenu);
p ← p.suivant;
FinTantque
retourner codage;
fin
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand


Chap. 7 : Application
Transmission
Deux choix sont possibles pour la transmission :
• soit l'on transmet une séquence correspondant au codage retenu tel que 0(a)10(b)1100(d)1101(c)111(r);
• soit on ne transmet que les fréquences et il reste à la charge du destinataire de reconstruire le code;
Olivier Raynaud
Université Blaise Pascal
Clermont-Ferrand